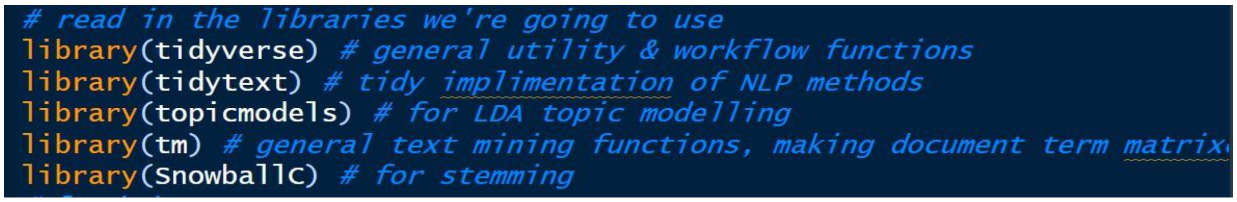
Topic Modeling for Twitter Dataset

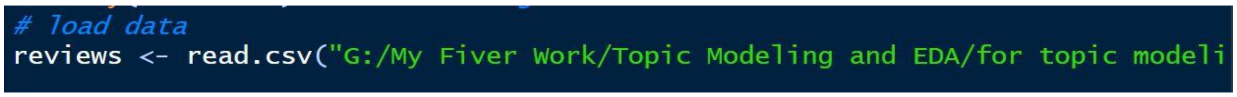
Topic modeling: The NLP task of identifying automatically identifying major themes in a text, usually by identifying informative words.

Before we can get started on topic modelling, we need to get our environment set up. First, let's load in the packages we're going to use. Here, I've added a comment letting you know why we need each package.



```
# read in the libraries we're going to use
library(tidyverse) # general utility & workflow functions
library(tidytext) # tidy implimentation of NLP methods
library(topicmodels) # for LDA topic modelling
library(tm) # general text mining functions, making document term matrix
library(SnowballC) # for stemming
```

And now load the dataset of twitterdata.csv

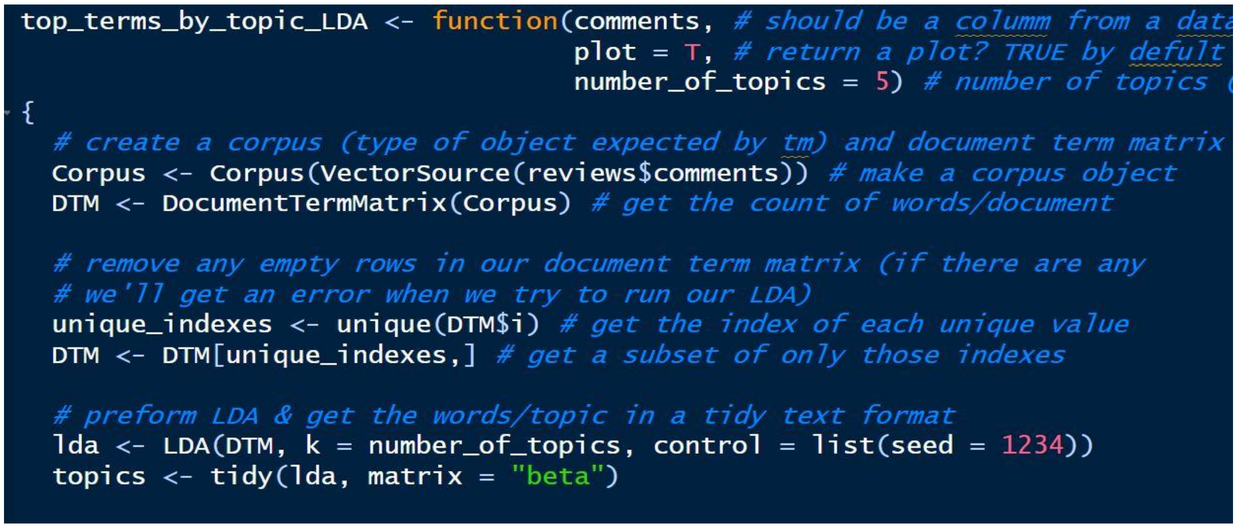


```
# load data
reviews <- read.csv("G:/My Fiver Work/Topic Modeling and EDA/for topic modeli
```

Now , using the LDA (Latent Dirichlet Allocation) Model, we can make a function to get and plot the most informative term by a specified number of topics.

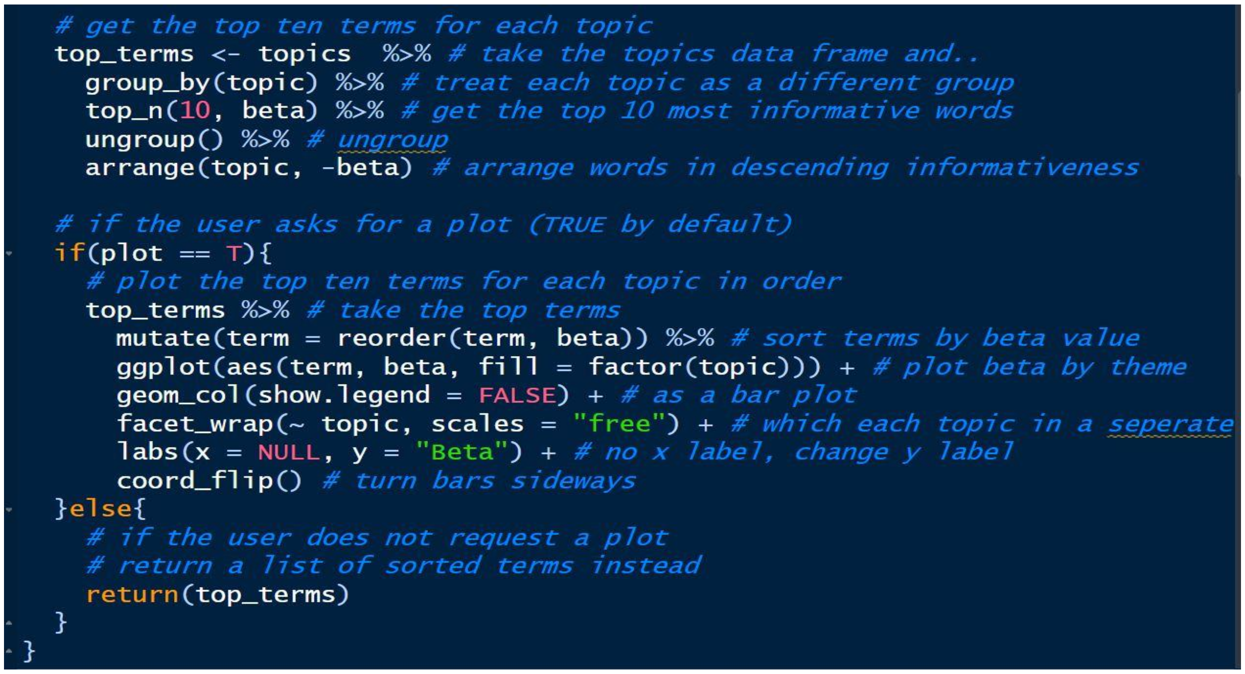
Unsupervised topic modeling with LDA :

So , the unsupervised topic modeling using LDA will be:



```
top_terms_by_topic_LDA <- function(comments, # should be a columm from a data
                                   plot = T, # return a plot? TRUE by defult
                                   number_of_topics = 5) # number of topics (
{
  # create a corpus (type of object expected by tm) and document term matrix
  Corpus <- Corpus(VectorSource(reviews$comments)) # make a corpus object
  DTM <- DocumentTermMatrix(Corpus) # get the count of words/document

  # remove any empty rows in our document term matrix (if there are any
  # we'll get an error when we try to run our LDA)
  unique_indexes <- unique(DTM$i) # get the index of each unique value
  DTM <- DTM[unique_indexes,] # get a subset of only those indexes

  # preform LDA & get the words/topic in a tidy text format
  lda <- LDA(DTM, k = number_of_topics, control = list(seed = 1234))
  topics <- tidy(lda, matrix = "beta")
```

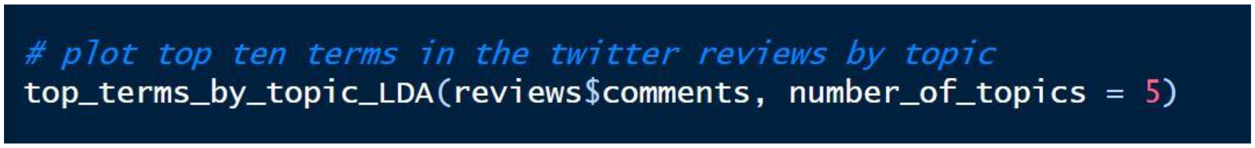
Setting up the corpus , we can store all the user comments that have been on the dataset.



```r
  # get the top ten terms for each topic
  top_terms <- topics  %>% # take the topics data frame and..
    group_by(topic) %>% # treat each topic as a different group
    top_n(10, beta) %>% # get the top 10 most informative words
    ungroup() %>% # ungroup
    arrange(topic, -beta) # arrange words in descending informativeness

  # if the user asks for a plot (TRUE by default)
  if(plot == T){
    # plot the top ten terms for each topic in order
    top_terms %>% # take the top terms
      mutate(term = reorder(term, beta)) %>% # sort terms by beta value
      ggplot(aes(term, beta, fill = factor(topic))) + # plot beta by theme
      geom_col(show.legend = FALSE) + # as a bar plot
      facet_wrap(~ topic, scales = "free") + # which each topic in a seperate
      labs(x = NULL, y = "Beta") + # no x label, change y label
      coord_flip() # turn bars sideways
  }else{
    # if the user does not request a plot
    # return a list of sorted terms instead
    return(top_terms)
  }
}
```

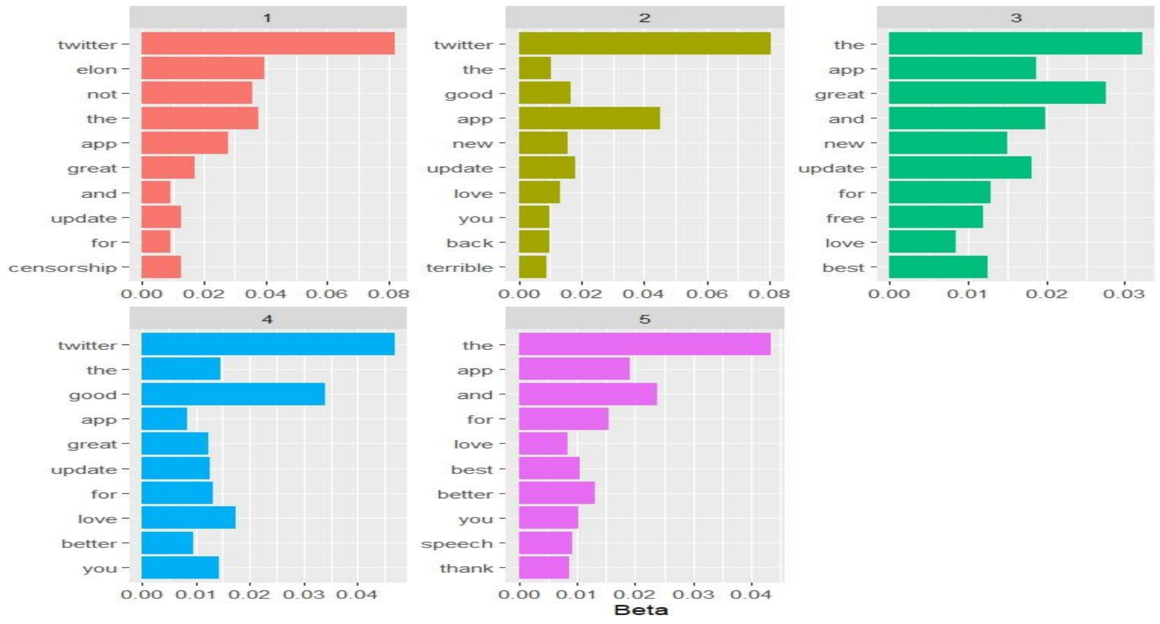
And finally building up the unsupervised LDA model to plot the specified word for the twitter data set.

Now after building the model, we plot the most 10 informative words for the topic modeling.



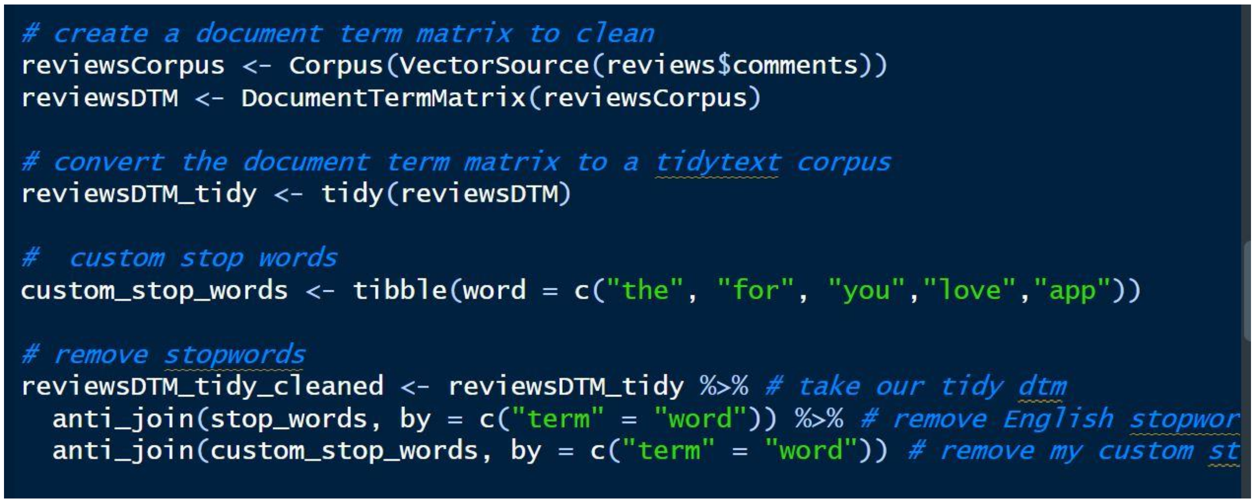
```r
# plot top ten terms in the twitter reviews by topic
top_terms_by_topic_LDA(reviews$comments, number_of_topics = 5)
```

And the output will be :

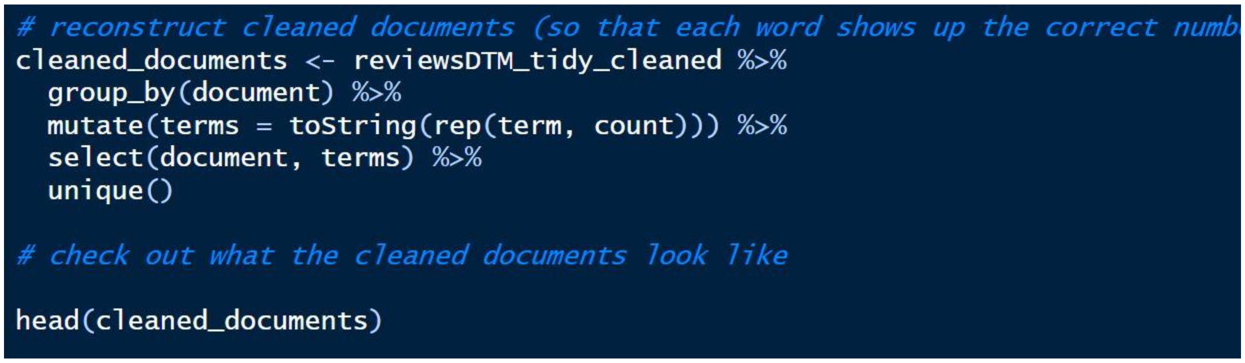


From the unsupervised LDA model it predicts some user comment that can be aliened with the user correlation between security and privacy updates and user satisfaction.

So, we can see that many unnecessary words ('for', 'you', 'and') are frequently occurred. So, we need to clean our corpus data for get better results.



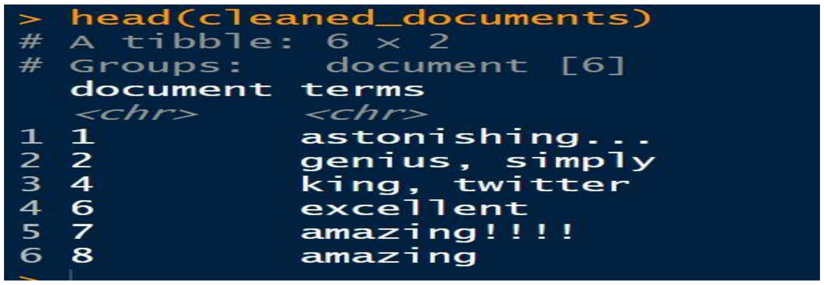
```
# create a document term matrix to clean
reviewsCorpus <- Corpus(VectorSource(reviews$comments))
reviewsDTM <- DocumentTermMatrix(reviewsCorpus)

# convert the document term matrix to a tidytext corpus
reviewsDTM_tidy <- tidy(reviewsDTM)

#  custom stop words
custom_stop_words <- tibble(word = c("the", "for", "you","love","app"))

# remove stopwords
reviewsDTM_tidy_cleaned <- reviewsDTM_tidy %>% # take our tidy dtm
  anti_join(stop_words, by = c("term" = "word")) %>% # remove English stopwor
  anti_join(custom_stop_words, by = c("term" = "word")) # remove my custom st
```



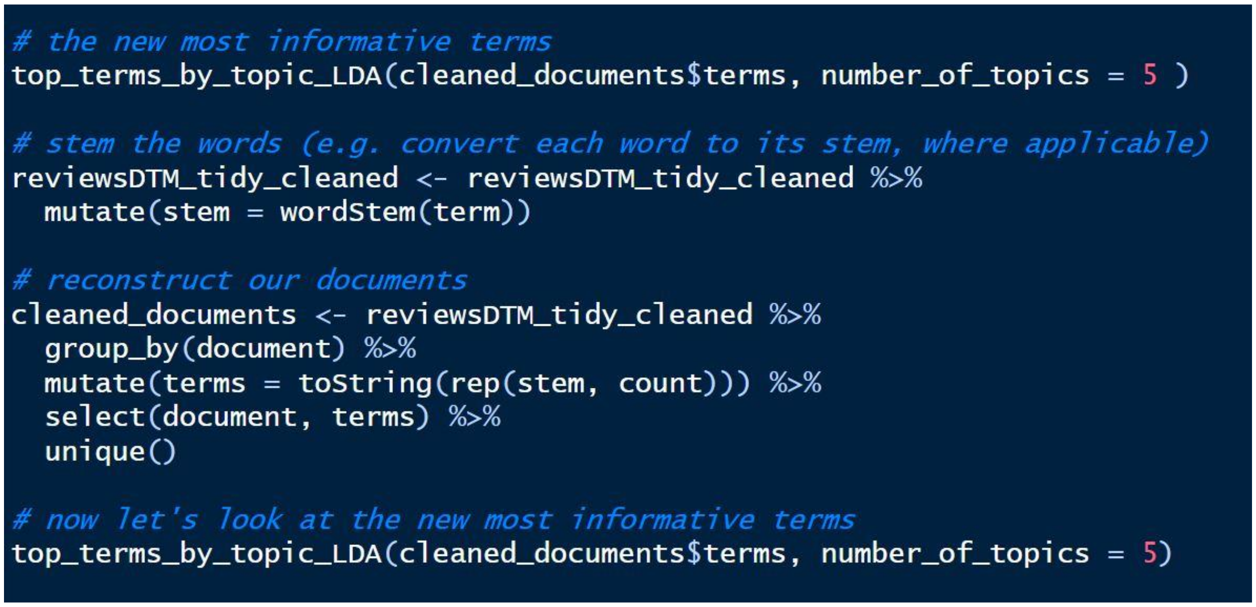
```
# reconstruct cleaned documents (so that each word shows up the correct numb
cleaned_documents <- reviewsDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(term, count))) %>%
  select(document, terms) %>%
  unique()

# check out what the cleaned documents look like

head(cleaned_documents)
```

So, we create a document term matrix (dtm) to clean up all the unnecessary words which not relevant to the analysis. And also create a cleaned documents that can give us the topic for modeling.

Some out put of cleaned corpus is:

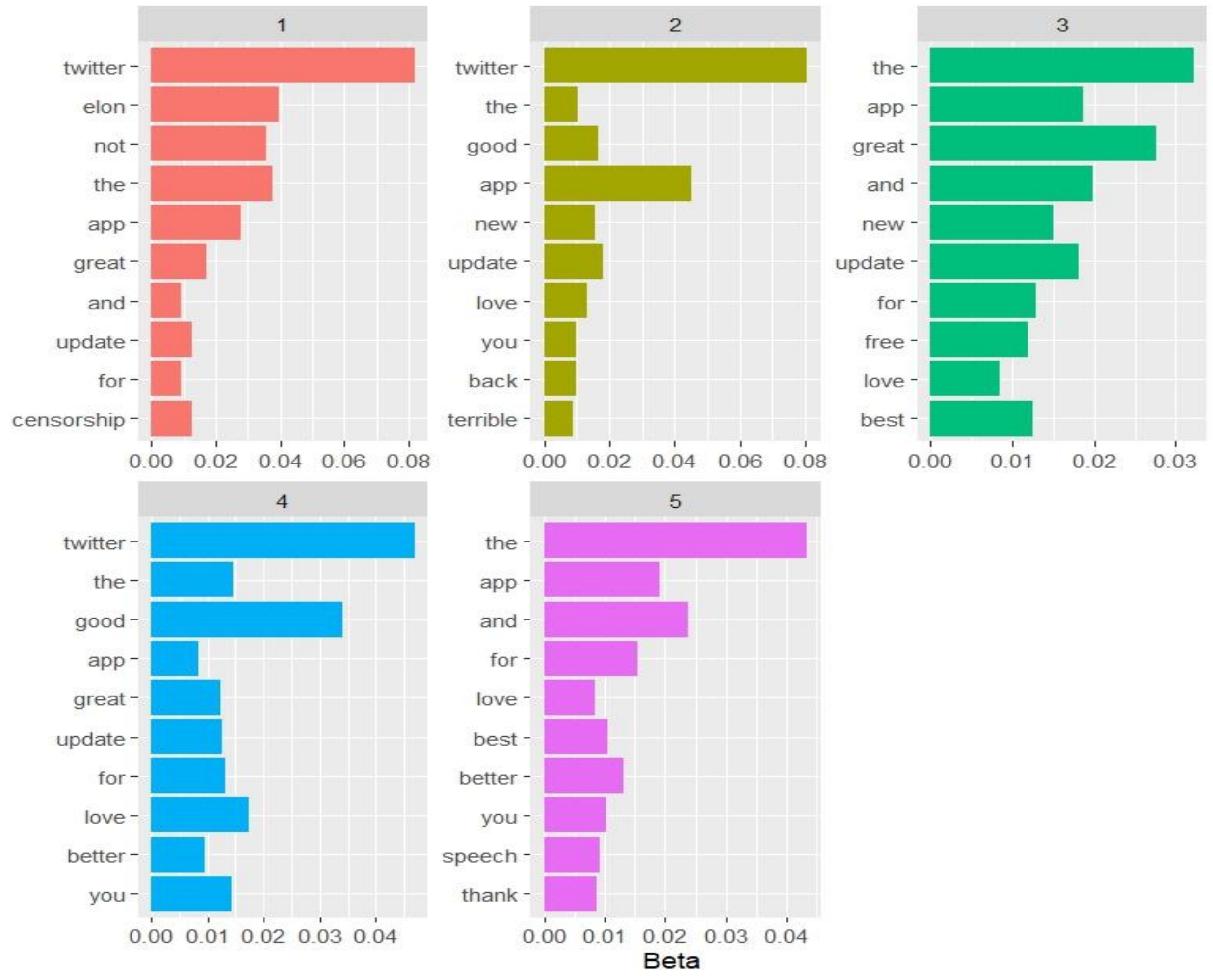


```
> head(cleaned_documents)
# A tibble: 6 x 2
# Groups:   document [6]
  document terms
  <chr>    <chr>
1 1        astonishing...
2 2        genius, simply
3 4        king, twitter
4 6        excellent
5 7        amazing!!!!
6 8        amazing
>
```



```
# the new most informative terms
top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics = 5 )

# stem the words (e.g. convert each word to its stem, where applicable)
reviewsDTM_tidy_cleaned <- reviewsDTM_tidy_cleaned %>%
  mutate(stem = wordStem(term))

# reconstruct our documents
cleaned_documents <- reviewsDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(stem, count))) %>%
  select(document, terms) %>%
  unique()

# now let's look at the new most informative terms
top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics = 5)
```

After cleaning the corpus, we have created a new informative term of the dataset.



From the new output we can see that it still generated some unnecessary words that might not be helpful for our prediction of user satisfaction and security and privacy update.

So, for better understanding we can used Supervised topic modeling with TF-IDF.

Supervised topic modeling with TF-IDF :

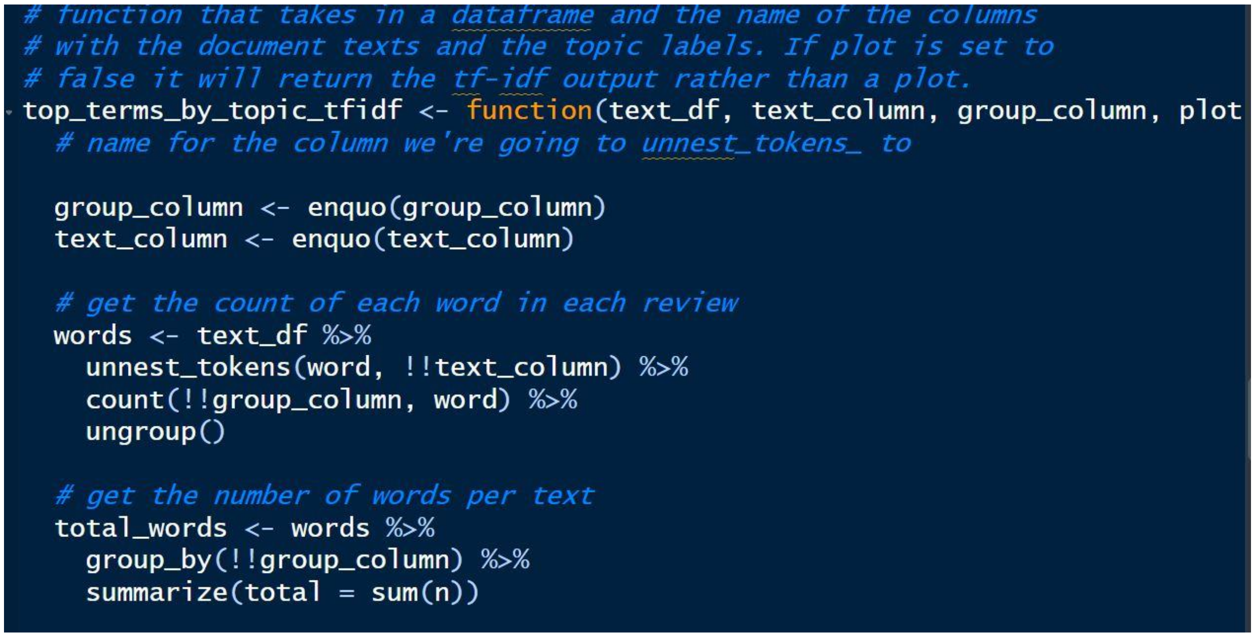
Now that we've given unsupervised topic modelling a go, let's try supervised topic modeling. For this, we're going to use something call TF-IDF, which stands for "term frequency-inverse document frequency".

The general idea behind how TF-IDF works is this:

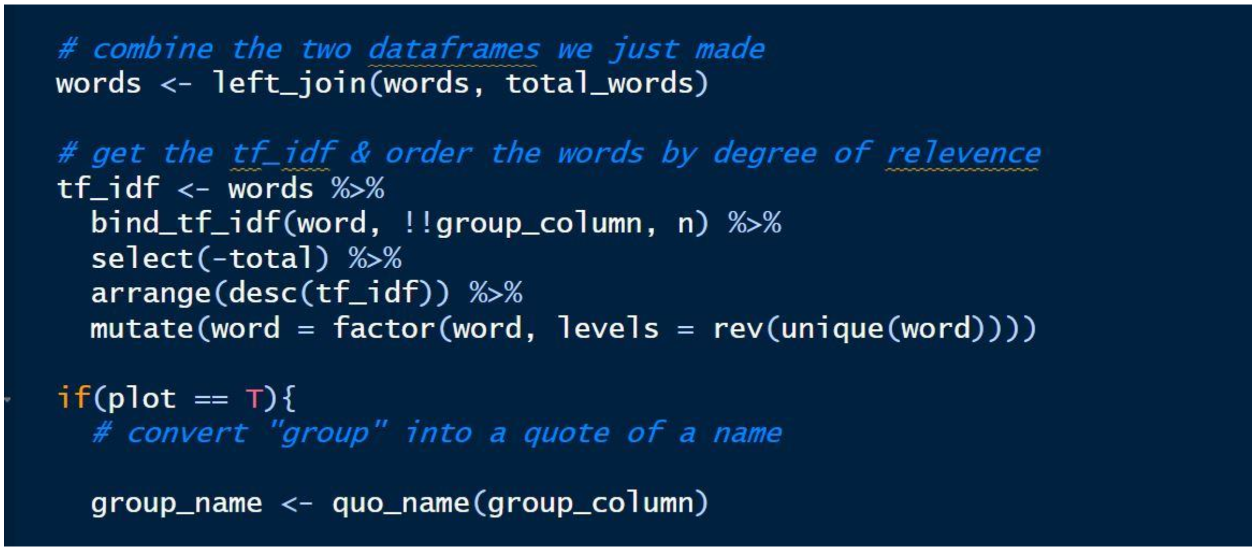
- Words that are very common in a specific document are probably important to the topic of that document
- Words that are very common in all documents probably aren't important to the topics of any of them

So, a term will receive a high weight if it's common in a specific document and also uncommon across all documents.

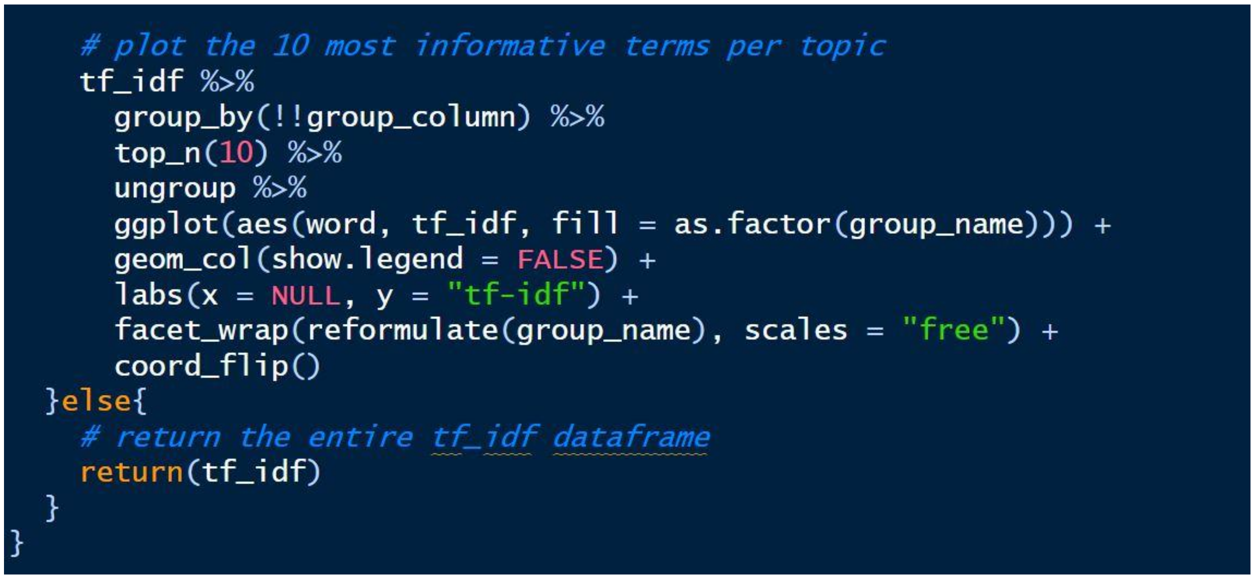
In order to streamline our analysis, we are writing a function that takes in a data frame, the name of the column that has the texts in it and the name of the column that has the topic labels in it.



```
# function that takes in a dataframe and the name of the columns
# with the document texts and the topic labels. If plot is set to
# false it will return the tf-idf output rather than a plot.
top_terms_by_topic_tfidf <- function(text_df, text_column, group_column, plot
  # name for the column we're going to unnest_tokens_ to

  group_column <- enquo(group_column)
  text_column <- enquo(text_column)

  # get the count of each word in each review
  words <- text_df %>%
    unnest_tokens(word, !!text_column) %>%
    count(!!group_column, word) %>%
    ungroup()

  # get the number of words per text
  total_words <- words %>%
    group_by(!!group_column) %>%
    summarize(total = sum(n))
```

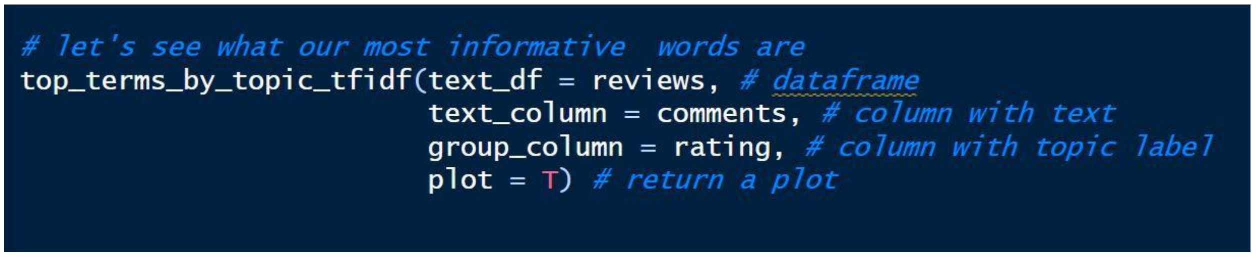


```r
  # combine the two dataframes we just made
  words <- left_join(words, total_words)

  # get the tf_idf & order the words by degree of relevence
  tf_idf <- words %>%
    bind_tf_idf(word, !!group_column, n) %>%
    select(-total) %>%
    arrange(desc(tf_idf)) %>%
    mutate(word = factor(word, levels = rev(unique(word))))

  if(plot == T){
    # convert "group" into a quote of a name

    group_name <- quo_name(group_column)
```



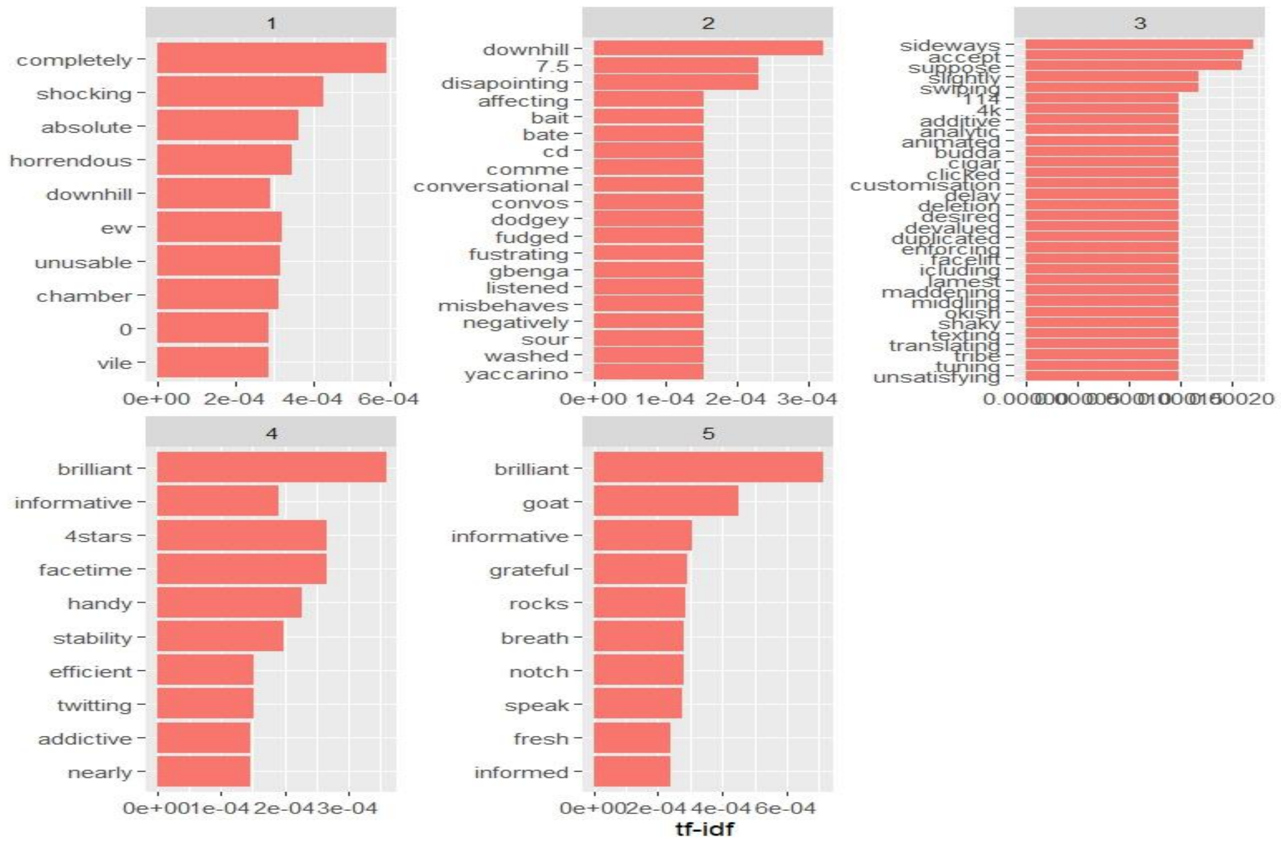
```r
    # plot the 10 most informative terms per topic
    tf_idf %>%
      group_by(!!group_column) %>%
      top_n(10) %>%
      ungroup %>%
      ggplot(aes(word, tf_idf, fill = as.factor(group_name))) +
      geom_col(show.legend = FALSE) +
      labs(x = NULL, y = "tf-idf") +
      facet_wrap(reformulate(group_name), scales = "free") +
      coord_flip()
  }else{
    # return the entire tf_idf dataframe
    return(tf_idf)
  }
}
```

So, we have built the function of supervised model. Now let's check the most informative words that generated with TF-IDF



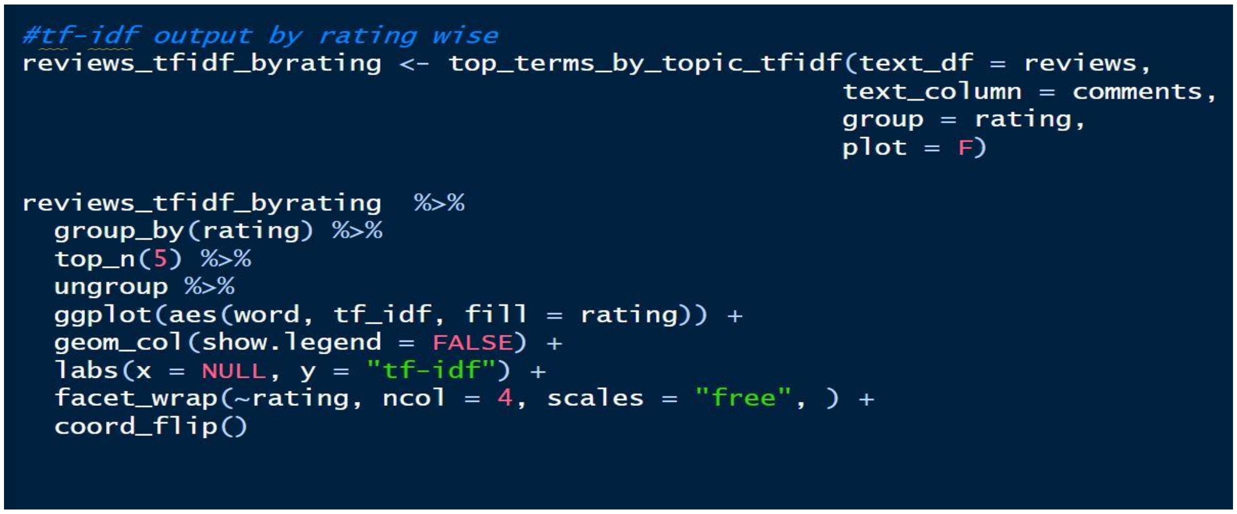
```r
# let's see what our most informative  words are
top_terms_by_topic_tfidf(text_df = reviews, # dataframe
                         text_column = comments, # column with text
                         group_column = rating, # column with topic label
                         plot = T) # return a plot
```

The output result of the supervised TF-IDF model is:



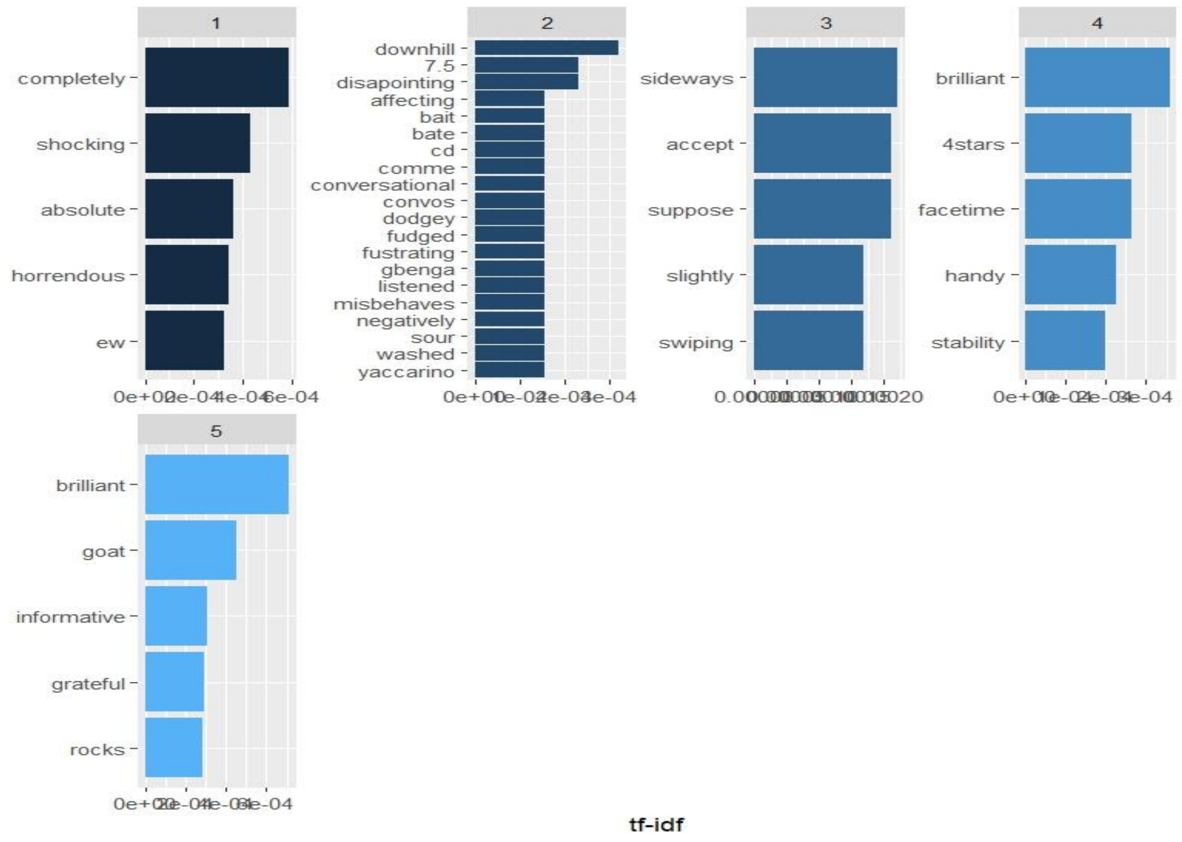
From the out we can see that we have unique words that can determine the user behavior pattern.

From this, we can see that negative reviews include words like "unusable", "disappointing", "unsatisfying" and "delay", while positive reviews also come up (both "brilliant" and "informative" show up in the top ten words).


```r
#tf-idf output by rating wise
reviews_tfidf_byrating <- top_terms_by_topic_tfidf(text_df = reviews,
                                                   text_column = comments,
                                                   group = rating,
                                                   plot = F)

reviews_tfidf_byrating  %>%
  group_by(rating) %>%
  top_n(5) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf, fill = rating)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~rating, ncol = 4, scales = "free", ) +
  coord_flip()
```

let's check out the comments for each rating. We are going to use the function above to return just the TF-IDF output and then plotting it by rating.

And the final output of using the reviews by ratings is:



From this, we can see that there are fewer negative reviews words while positive reviews words are showing positive user experience(both "brilliant" ,"informative" and ‘stability’ show up in the top five words).

Conclusion:

From overall analysis we can conclude that , the twitter datasets provide us some insides about user satisfaction level but not corelated with the user security and privacy updates.