Classification of hamspam data using quanteda package

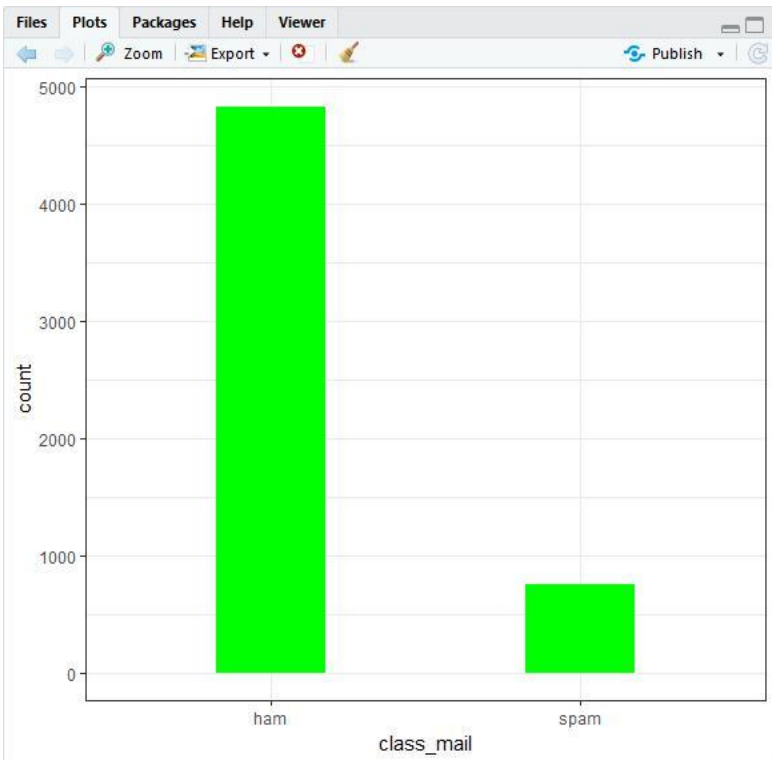
Quanteda Package of R :

Step 1: Loading the data and assigning TAG/Label to the data. Counts of Spam and Ham counts

```
> table(sms_data$class_mail)

 ham spam
4827  747
> 
```

Distribution using ggplot



Step 2: Creating the corpus: corpus() constructs a corpus class object.

Step 3: Preprocessing the corpus

Normalization: Lowercasing and stemming, Removing stopwords, Numbers, Symbols

Data Look: Example

```
                                               comments
634
        Oh that was a forwarded message. I thought you send that to me
3469                     URGENT! We are trying to contact U. Todays draw shows that you have won a £800 prize
GUARANTEED. Call 09050001808 from land line. Claim M95. Valid12hrs only
3395
                                                                         Ok thanx...
3473 Heart is empty without love.. Mind is empty without wisdom.. Eyes r empty without dreams &amp; Life is em
pty without frnds.. So Alwys Be In Touch. Good night &amp; sweet dreams
4796
                              Sorry that was my uncle. I.ll keep in touch
3566
                          Do you always celebrate NY's with your family ?
> |
```

Code: dfm

Creating a Document-Feature Matrix

Construct a sparse document-feature matrix, from a character, corpus, tokens.

```
##Pre-Processing the corpus dfm()
main_corpus <- dfm(corpus_comments, tolower = TRUE, remove_punct = TRUE, remove_twitter = TRUE,remove_symbols = T
```

STEP 4: Filtering and weighting

On the Document Frequency Matrix: doc_freq

Computing the (Weighted) Document Frequency of a Feature. This returns a (weighted) document frequency for each term.

The default threshold is zero, meaning that any feature occurring at least once in a document will be counted

Filtering the terms where docfreq >=2

dfm_weight: Weight The Feature Frequencies in a Dfm. Using dfm_tfidf(main_corpus)

Returns a document by feature matrix with the feature frequencies weighted according to "tfidf"

sparsity: sparsity(main_corpus) : Compute the sparsity of a document-feature matrix

By keep only words occurring >= 10 times and in >= 2 documents

```
sparsity(dfm_trim(main_corpus, min_termfreq = 10, min_docfreq = 2))
sparsity(main_corpus)
```

dfm_trim:

Trimming a Dfm Using Frequency Threshold-Based Feature Selection

Returns a document by feature matrix reduced in size based on document and term frequency, usually in terms of a minimum frequency, but may also be in terms of maximum frequencies.

```
> sparsity(dfm_trim(main_corpus, min_termfreq = 10, min_docfreq = 2))
[1] 0.9984101
```

dfm_compress: Compress a dfm by combining identical elements.

Recombine A Dfm By Combining Identical Dimension Elements

"Compresses" or groups a dfm whose dimension names are the same, for either documents or features.

```
> dfm_compress(main_corpus, margin = c("both", "documents", "features"))
Document-feature matrix of: 5,574 documents, 3,580 features (99.8% sparse).
```

Below generating top 50 features having min count of 10 and appearing in 5 of the documents

Code:

```
##To see the features
main_corpus_trim = dfm_trim(main_corpus,min_count=10,min_docfreq = 5)
##generating top 50 features having minm count of 10 and
#appearing in 5 of the documents
topfeatures(main_corpus_trim,n=50)
```

Output:

```
> topfeatures(main_corpus_trim,n=50)
      ur     call       gt       lt     free     love     good      day     time        ü     text     send 
615.6437 597.0340 590.2086 586.4966 538.7159 469.2451 467.0253 455.2757 445.1717 443.8559 435.8675 417.3380 
    stop      txt      lor       da     home    reply     back      pls     dont   mobile    today     week 
410.8899 408.0214 406.8772 392.1541 386.2401 385.3403 381.3932 380.4114 378.5830 376.9562 367.8140 355.3636 
   happy     dear    night    phone    great    claim      hey      msg     hope     give      amp     make 
354.4714 347.7867 343.2182 342.9004 335.9973 331.4843 324.0015 321.6068 321.1773 317.2212 315.9797 312.5649 
    work      wat    prize   number     life       im  message tomorrow     yeah    nokia     miss     meet 
311.0616 309.6118 305.3733 301.8334 300.6620 295.6323 295.3920 295.3920 285.5464 284.3879 281.8480 280.2022 
    babe  morning 
273.0809 269.5798 
>
```

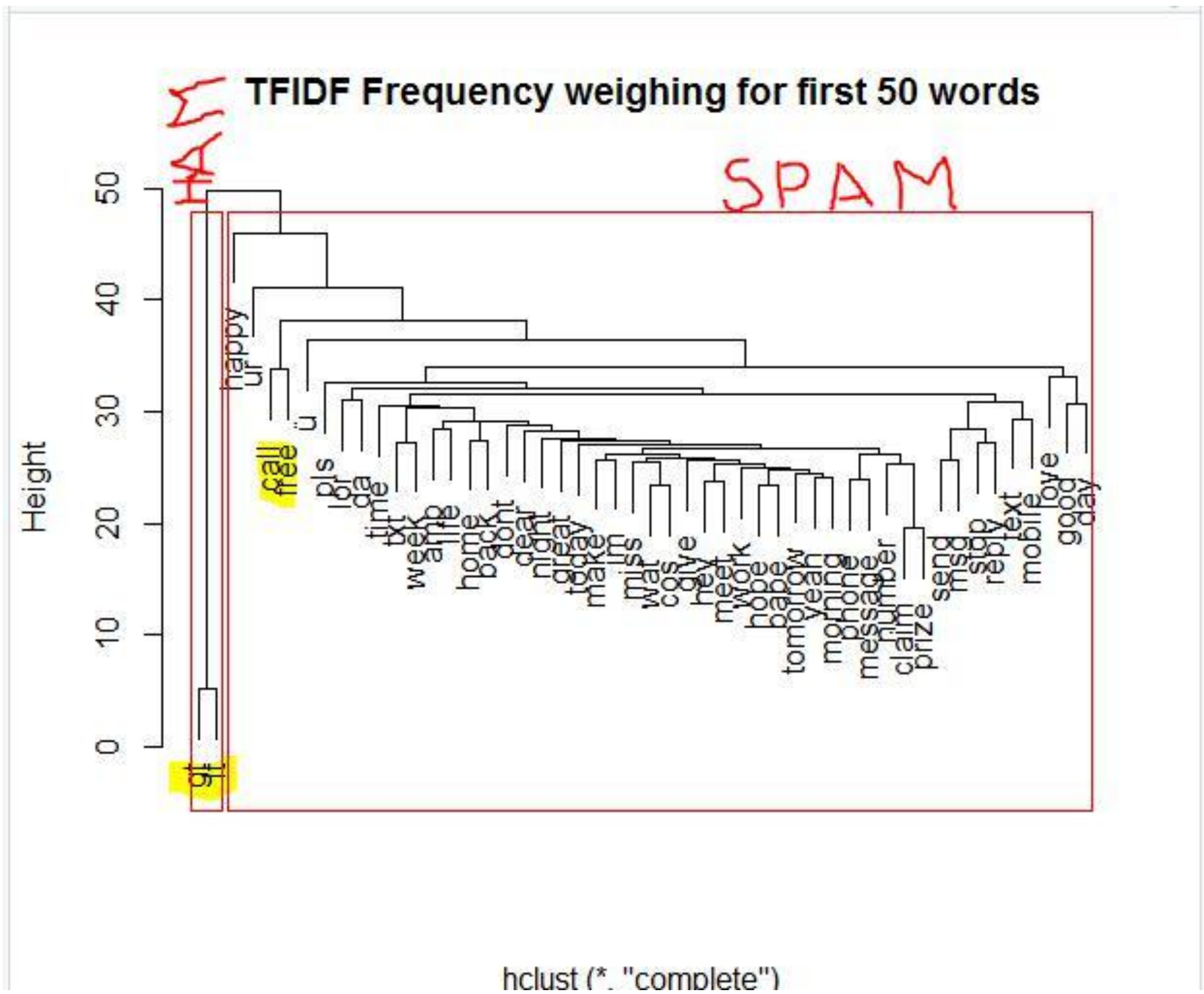
- ✓ Here in weighing the **terms**, term **frequency-inverse document frequency (tf-idf),** this down-weights terms that occur in many documents in the corpus.
- ✓ Using a document frequency threshold and weighting can be performed on a **DTM.**
- ✓ **quanteda** includes the functions **docfreq, tf, and tfidf**, for obtaining **document frequency, term frequency, and tf-idf respectively.**

Step 5: Creating a cluster for the top 50 words

dist: Matrix Distance/Similarity Computation:

dist functions compute and return the auto-distance/similarity matrix between either rows or columns of a matrix/data frame and as well as the cross-distance matrix between two matrices/data frames/lists

Dendobram:



Step 6: Generating wordcloud:

Text mining methods allow us to highlight the most frequently used keywords in a paragraph of texts. One can create a word cloud, also referred as text cloud or tag cloud, which is a visual representation of text data. Most frequent words appears larger in the worcloud.

Quanteda's corpus() command is used to construct a corpus from the Text field of our raw data.

A corpus can be thought of as a master copy of our dataset from which we can pull subsets or observations as needed.

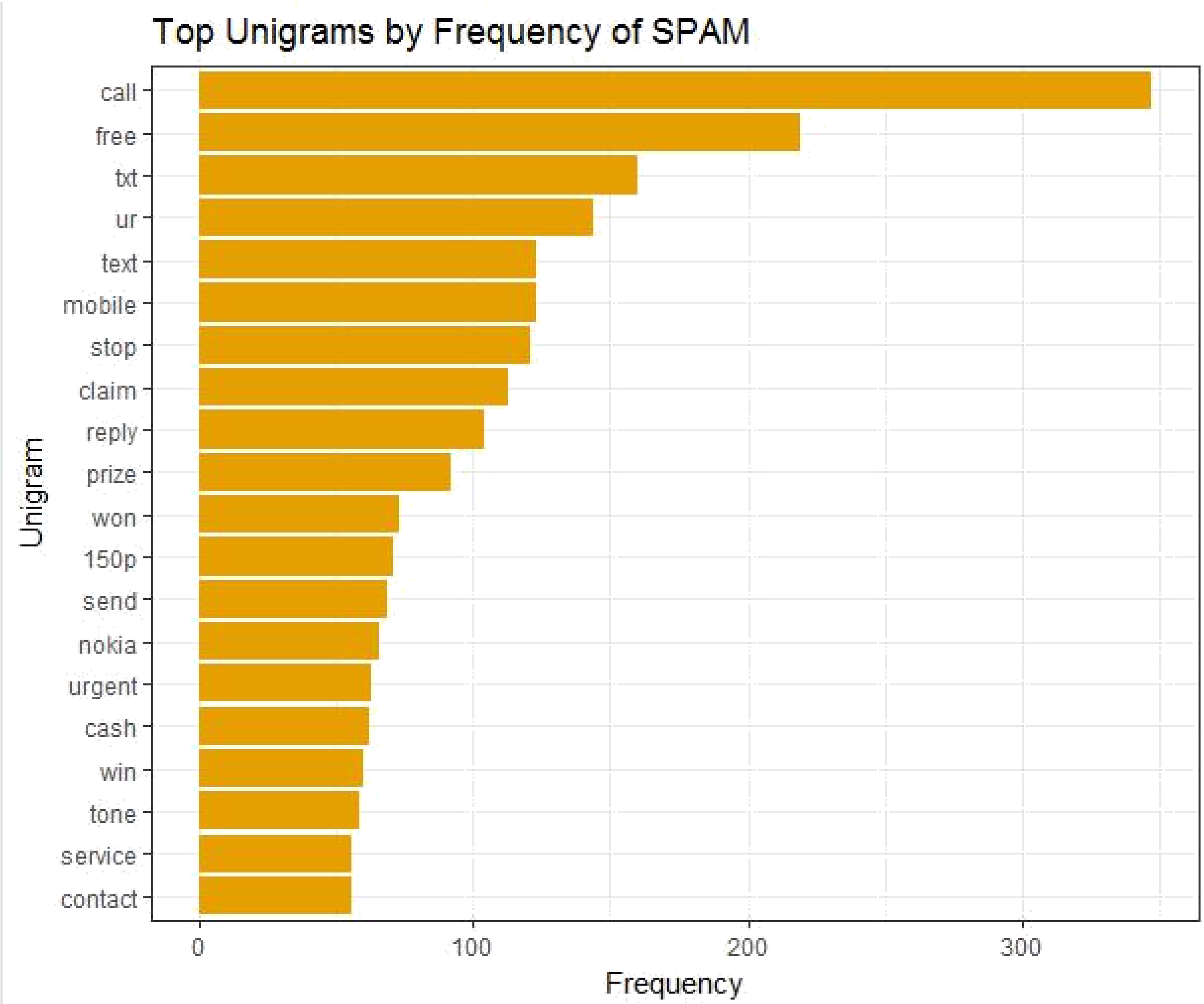
textplot_wordcloud

Plot Features As A Wordcloud

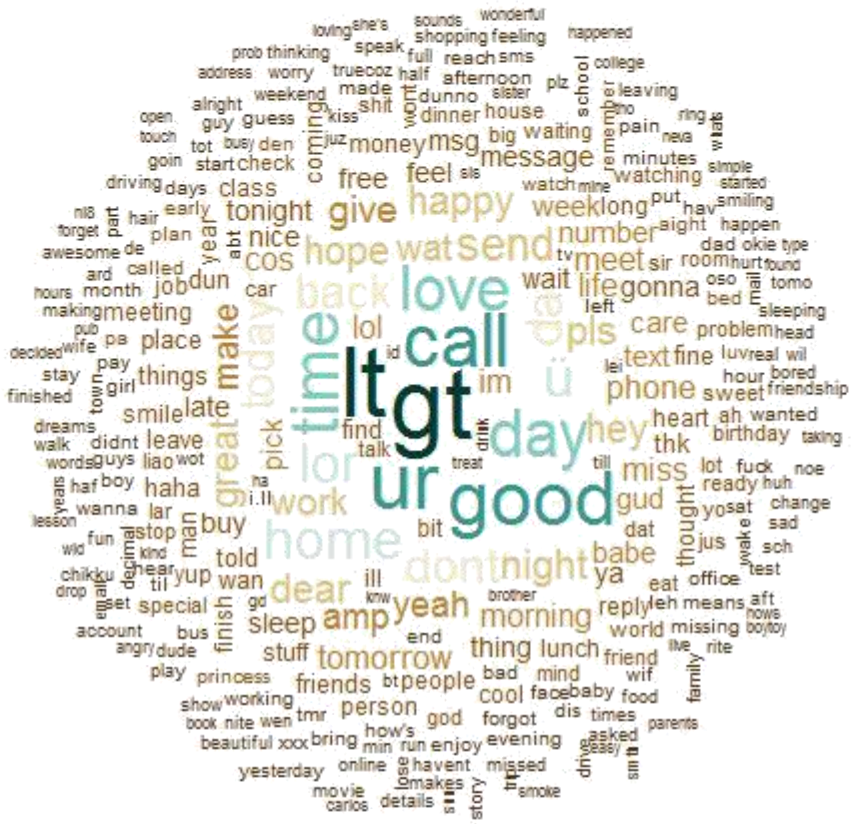
Plot a dfm object as a wordcloud, where the feature labels are plotted with their sizes proportional to their numerical values in the dfm. When comparison = TRUE, it plots comparison word clouds by document

Generating SPAM Worcloud: Free and call words are most frequently used in SPAM.

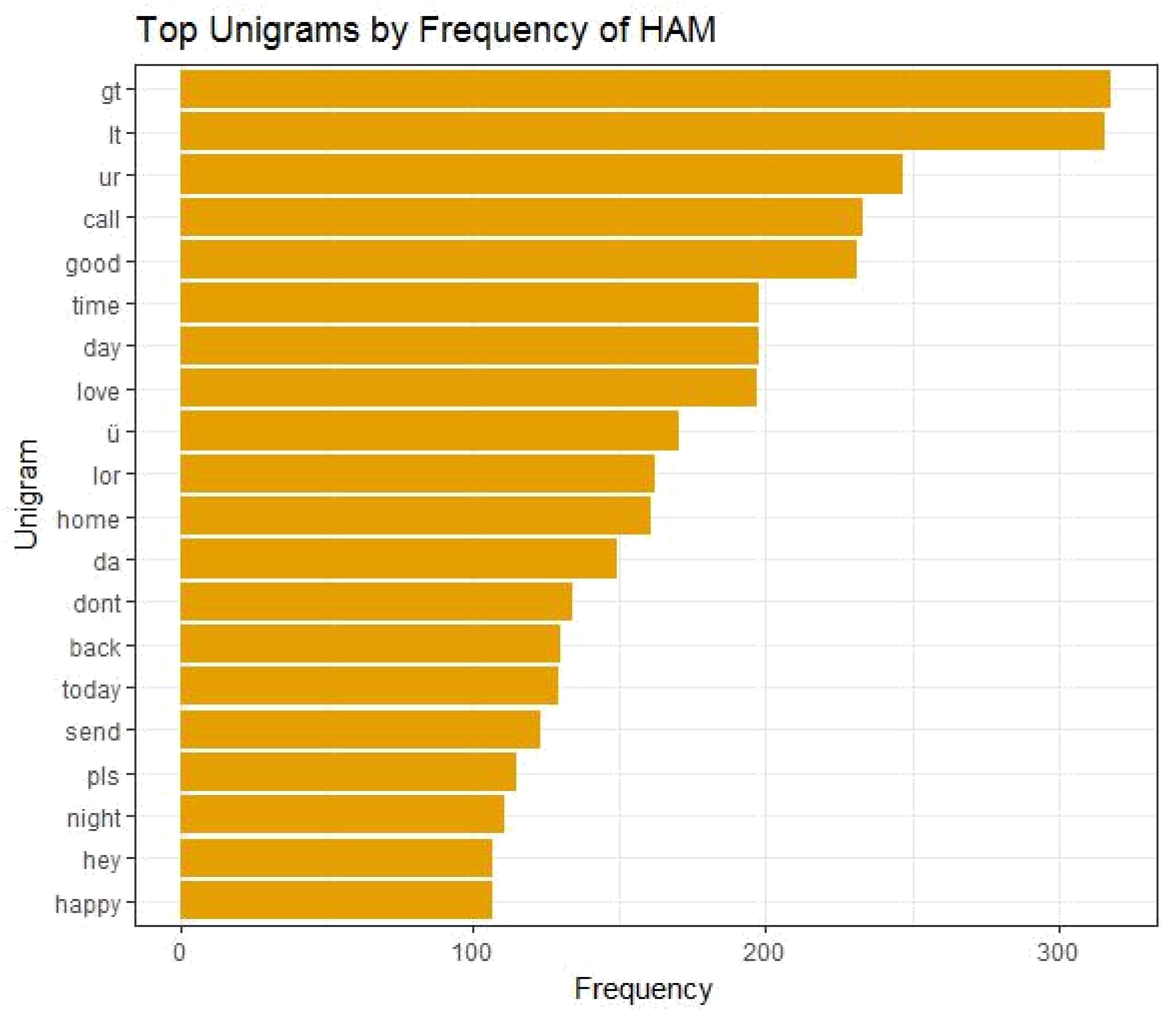
Showing Top 20 UNIGRAMS IN SPAM Wordcloud:



Generating HAM Worcloud: lt and gt words are most frequently used in HAM.



Showing Top 20 UNIGRAMS IN HAM Wordcloud:



STEP 7: NAÏVE BAYES CLASSIFICATION

Prediction Using Naive Bayes Classifier

Naive Bayes classifiers are a class of simple linear classifiers which are conditional probability models based on Bayes Theorem i.e.

P(Y∈Kj|Xi)=P(X1|Y).P(X2|Y)……P(Xi|Y). P(Y∈Kj)

where Xi are the number of inputs and Y is discrete response variable and Kj are the number of class labels.

Naive Bayes classifiers follow Conditional Independence Theorem i.e the features Xi are uncorrelated and independent of each other.

Secondly, they assume that the data samples are drawn from a identical and independent distribution.

The model outputs the Probabilities of the message being Spam or ham.

```
> class_table <- table(sms.predictions$nb.predicted, sms_data.test$class_mail)
> class_table

       ham spam
  ham  1467    6
  spam    7  192
> |
```

True Positive: Correctly predicted event values.

- **we must calculate the number of correct predictions for each class.**
- **HAM Classified as Ham = 1467**
- **Spam classified as Spam= 192**
- **Ham classified as SPAM = 7**
- **Spam classified as Ham = 6**

Confusion Matrix:

```
> confusionMatrix(class_table, mode = "everything")
Confusion Matrix and Statistics


       ham spam
  ham  1467    6
  spam    7  192

               Accuracy : 0.9922
                 95% CI : (0.9867, 0.9959)
    No Information Rate : 0.8816
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9628
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9953
            Specificity : 0.9697
         Pos Pred Value : 0.9959
         Neg Pred Value : 0.9648
              Precision : 0.9959
                 Recall : 0.9953
                     F1 : 0.9956
             Prevalence : 0.8816
         Detection Rate : 0.8774
   Detection Prevalence : 0.8810
      Balanced Accuracy : 0.9825

       'Positive' Class : ham

>
```

This Model has an Accuracy of 99.22%.

- **Sensitivity or Recall : the proportion of actual positive cases which are correctly identified. = 99.53%**
- **Specificity : the proportion of actual negative cases which are correctly identified.=96.97**