Toxic comment classification

Sankepally Vikram Reddy

17/04/2018

Contents:

Chapter 1:

Introduction:

Problem Statement

Toxic comment classification

In this project I was given to build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. You'll be using a dataset of comments from Wikipedia talk page edits.l

Data:
We have a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

training_data=159571 rows,8 columns

test_data=153164 rows,8 columns

<u>Warning:</u>

<u>As data is huge it took some hours for me to execute the process.So make sure your computer is fast enough(RAM) to execute the process.</u>

we must create a model which predicts a probability of each type of toxicity for each comment.

>head(training_data)

```
             id
1 0000997932d777bf
2 000103f0d9cfb60f
3 000113f07ec002fd
```

4 0001b41b1c6bb37
5 0001d958c54c6e35
6 00025465d4725e87

comment_text
1
Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27
2
D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)
3
Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info.
4 "\nMore\nI can't make any real suggestions on improvement - I wondered if the section statistics should be later on, or a subsection of ""types of accidents"" -I think the references may need tidying so that they are all in the exact same format ie date format etc. I can do that later on, if no-one else does first - if you have any preferences for formatting style on references or want to do it yourself please let me know.\n\nThere appears to be a backlog on articles for review so I guess there may be a delay until a reviewer turns up. It's listed in the relevant form eg Wikipedia:Good_article_nominations#Transport "
5
You, sir, are my hero. Any chance you remember what page that's on?
6
"\n\nCongratulations from me as well, use the tools well. \302\240\302\267 talk "

	toxic	severe_toxic	obscene	threat	insult	identity_hate
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Predictor Variables

- **Comment_text**

Target variables

- threat
- Severe_toxic
- Insult
- Identity_hate
- toxic
- obscene

Exploratory data analysis

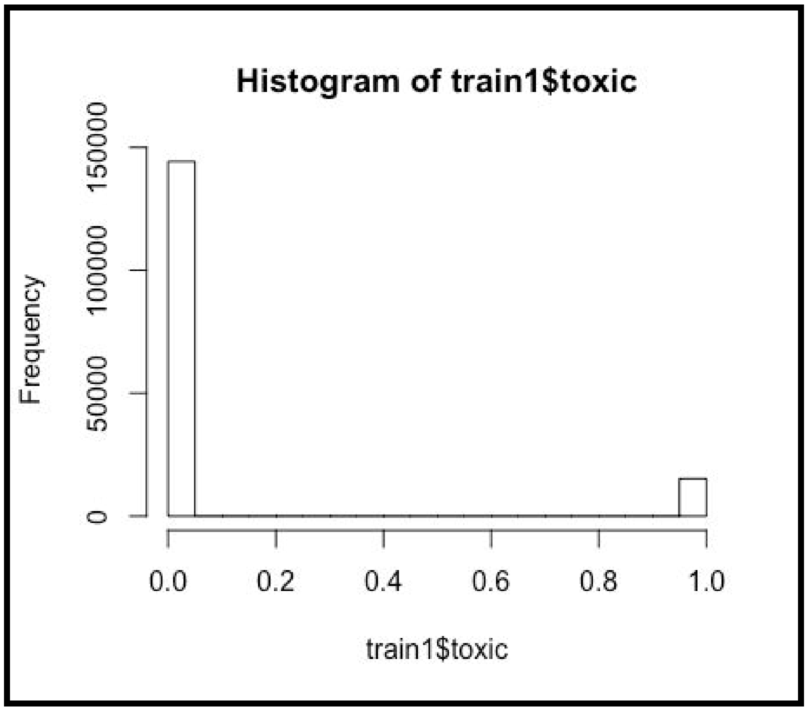
Steps I have done

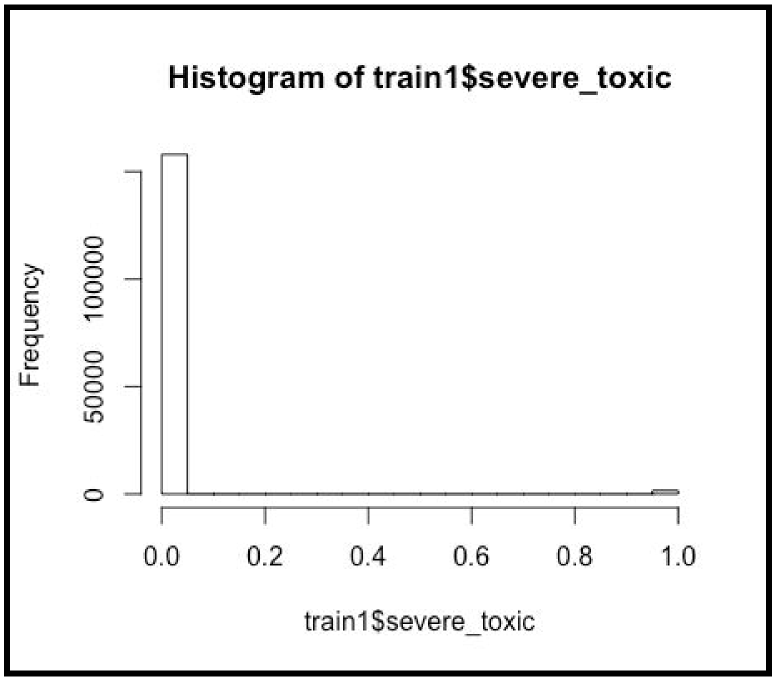
- Check percentage classes
- Class imbalance
- Plot histograms
- Corr plot of target variables
- Histogram of words

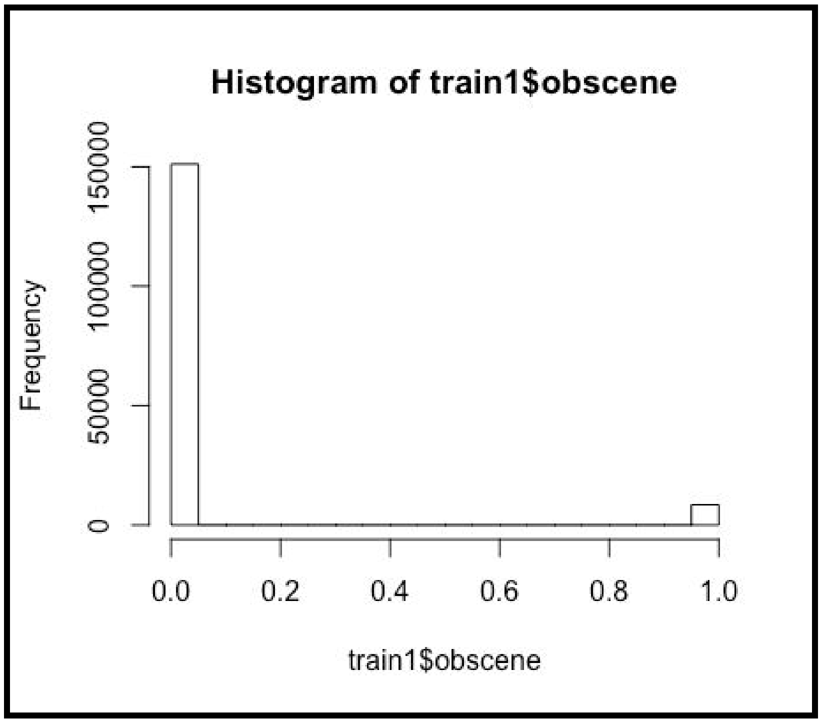
Distribution of classes

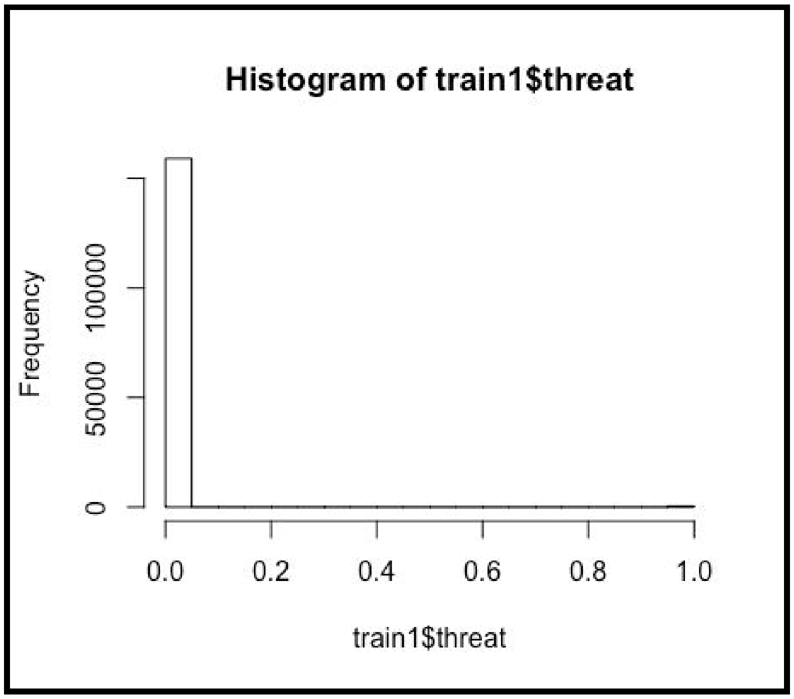
We have to check the percentage of classes in target variable to check if the data is imbalanced or not

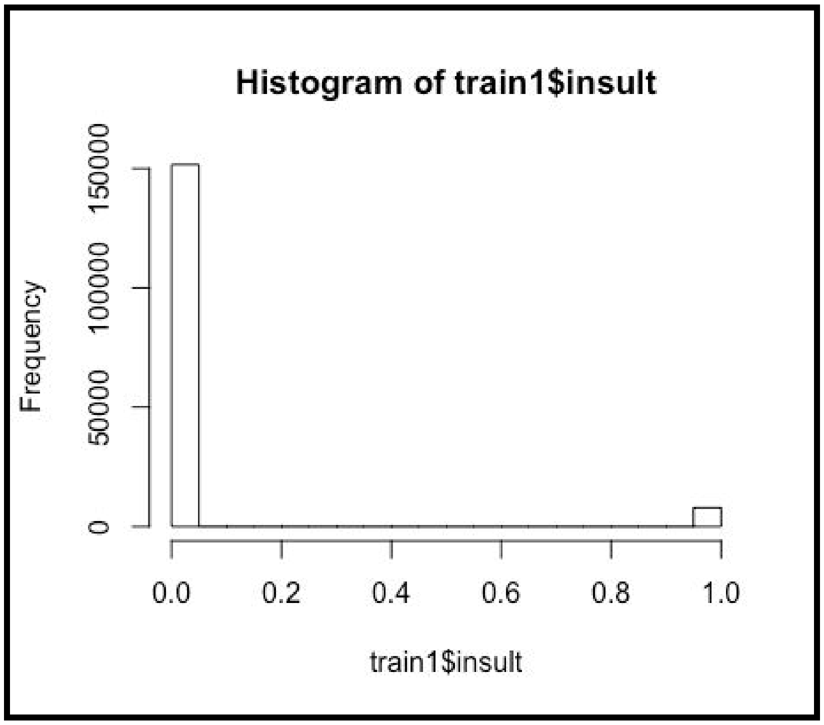
There are two classes in each of 6 target variables ,they are 0 and 1

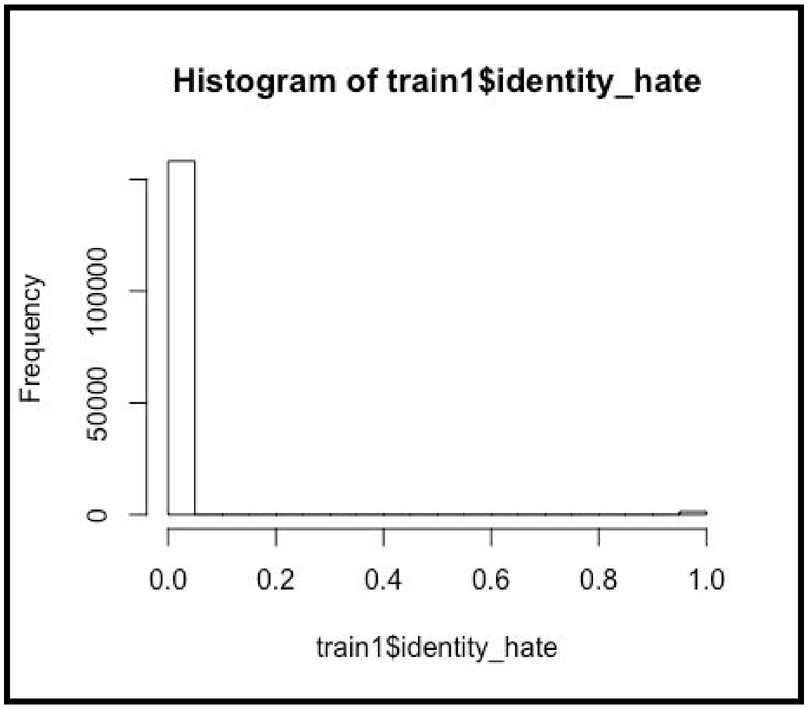

Histogram of train1$toxic
Frequency
0
50000
100000
150000
0.0
0.2
0.4
0.6
0.8
1.0
train1$toxic


Histogram of train1$severe_toxic
Frequency
0
50000
100000
0.0
0.2
0.4
0.6
0.8
1.0
train1$severe_toxic


Histogram of train1$obscene
Frequency
0
50000
100000
150000
0.0
0.2
0.4
0.6
0.8
1.0
train1$obscene


Histogram of train1$threat
Frequency
0
50000
100000
0.0
0.2
0.4
0.6
0.8
1.0
train1$threat





From the above figures we can say that there is a class imbalance problem

Therefore we have to necessary steps to overcome the problem of class imbalance

If we do not deal with this problem.even if the model predicts all column as 1 it will be of >95% accuracy.which is of no use

So we have to do necessary pre processing to overcome this problem

Dealing with class imbalance problem

There are many ways of dealing with class imbalance problem

- Oversampling
- Synthetic sampling
- Under sampling

This method works with majority class. It reduces the number of observations from majority class to make the data set balanced. This method is best to use when the data set is huge and reducing the number of training samples helps to improve run time and storage troubles.

One major drawback is

removing observations may cause the training data to lose important information pertaining to majority class.

2. Oversampling

This method works with minority class. It replicates the observations from minority class to balance the data. It is also known as upsampling. An advantage of using this method is that it leads to no information loss.

The disadvantage of using this method is that, since oversampling simply adds replicated observations in original data set, it does adding multiple observations of several types, thus leading to overfitting.

And more over, the training accuracy of such data set will be high, but the accuracy on unseen data will be worse.

3. Synthetic Data Generation

In simple words, instead of replicating and adding the observations from the minority class, it overcome imbalances by generates artificial data. It is also a type of oversampling technique.

In regards to synthetic data generation, synthetic minority oversampling technique (SMOTE) is a powerful and widely used method. SMOTE algorithm creates artificial data based on feature space (rather than data space) similarities from minority samples

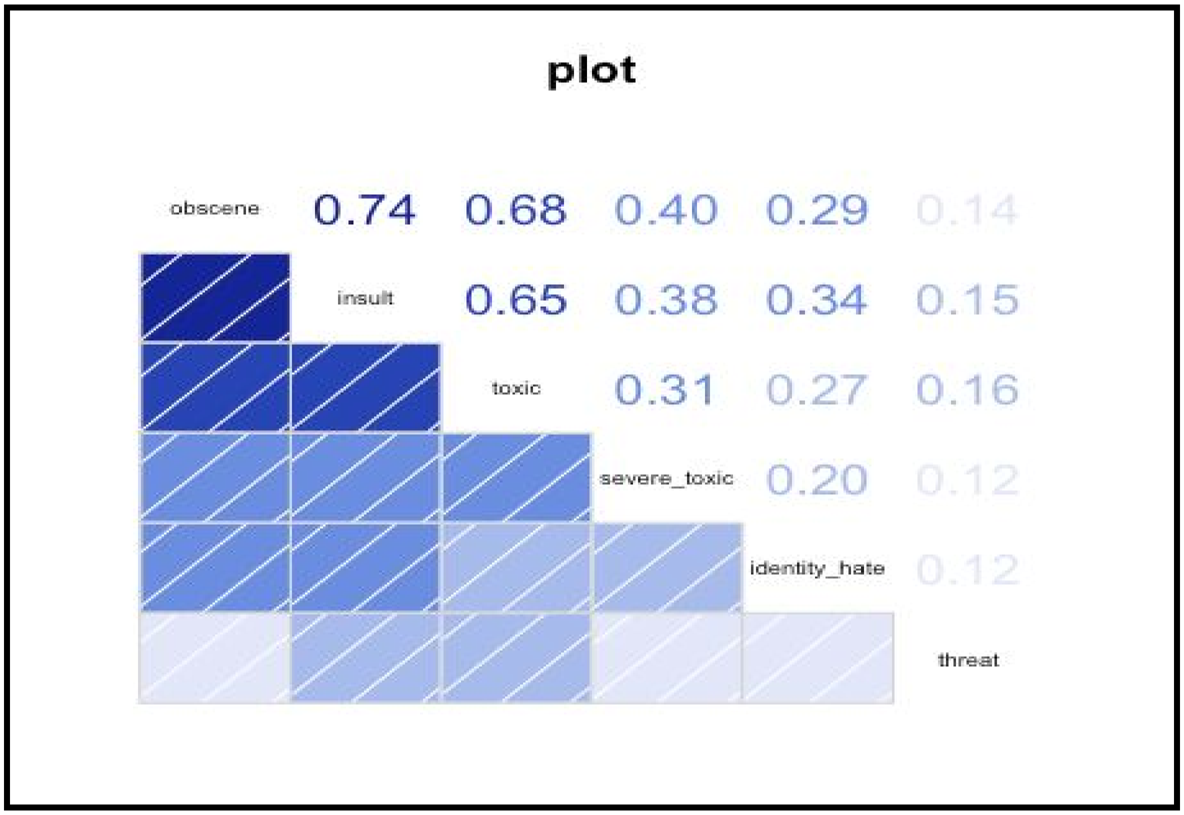
In order to generate artificial data, it uses bootstrapping and k-nearest neighbors.

As the data has huge number of columns over sampling and synthetic sampling will increase the number of observations and eventually the model will break down in process

So I have chosen downsampling .

I have tried doing with synthetic sampling but it got break down causing a fatal error in RStudio

3.Corr plot of target variables



The above plot shows that correlation between variables

it is very useful to highlight the most correlated variables in a data table. In this plot, **correlation coefficients** is colored according to the value.

Correlation matrix can be also reordered according to the degree of association between variables.

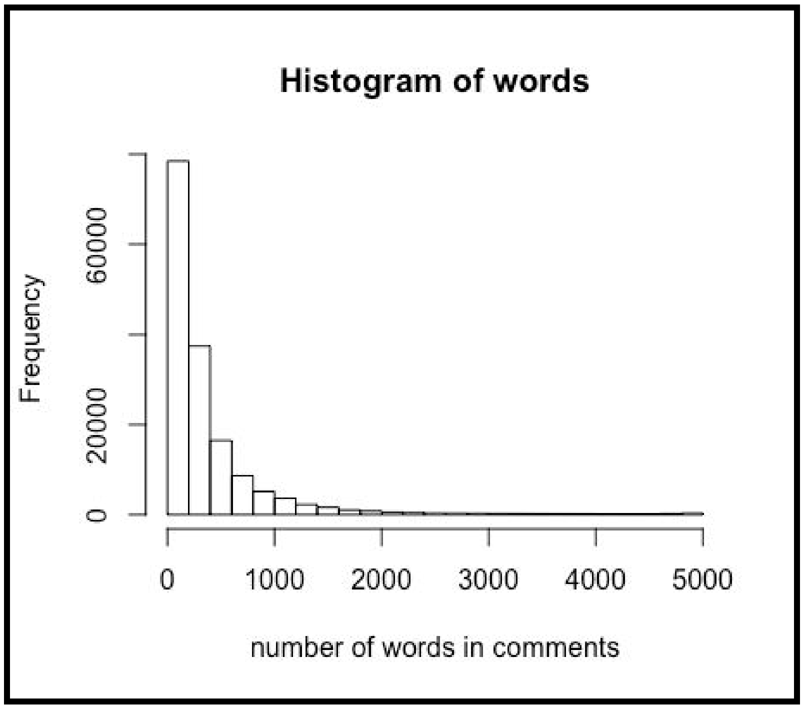
A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable in the table is correlated with each of the other values in the table . This allows you to see which pairs have the highest correlation.

Correlation matrix

	toxic	severe_toxic	obscene	threat	insult
toxic	1.0000000	0.3086191	0.6765145	0.1570584	0.6475181
severe_toxic	0.3086191	1.0000000	0.4030145	0.1236013	0.3758072
obscene	0.6765145	0.4030145	1.0000000	0.1411790	0.7412724
threat	0.1570584	0.1236013	0.1411790	1.0000000	0.1500224
insult	0.6475181	0.3758072	0.7412724	0.1500224	1.0000000
Identity_hate	0.2660094	0.2016002	0.2868669	0.1151283	0.3377363

	identity_hate
toxic	0.2660094
severe_toxic	0.2016002
obscene	0.2868669
threat	0.1151283
insult	0.3377363
identity_hate	1.0000000

Histogram of words



From the above plot we can say that most of the comments have total number of words in between 0 to 500

Chapter 2:

Pre Processing

The important step in designing a model is clean the data first before passing to the model.Preprocessing is an important task and critical step in Text mining, Natural Language Processing (NLP) and information retrieval (IR). In the area of Text Mining, data preprocessing used for extracting interesting and non-trivial and knowledge from unstructured text data. Information Retrieval (IR) is essentially a matter of deciding which documents in a collection should be retrieved to satisfy a user's need for information.

Unfortunately, the words that appear in documents and in queries often have many structural variants. So before the information retrieval from the documents, the data preprocessing techniques are applied on the target data set to reduce the size of the data set which will increase the effectiveness of IR System

The objective of this step is to analyze the issues of preprocessing methods such as Tokenization, Stop word removal and Stemming for the text documents Keywords: Text Mining, NLP, IR, Stemming

Steps

- Replace certain words
- Cleaning text comments
- Creating corpus words
- creating document term matrix

Replace certain words

As the unstructured data from wikipedia has certain words which we need to convert to their original meaning accurately

So I have replaced words with their original meaning

Such as

\'ve-I have

\'scuse=excuse

\'re=are

\'ll=I will

Cleaning text comments

Let's now build a corpus out of this vector of strings. A corpus is a collection of documents

Before that we have to clean the text comments.after that we have to create a corpus

Steps for cleaning unstructured data

- **tolower()**: Make all characters lowercase
- **removePunctuation()**: Remove all punctuation marks
- **removeNumbers()**: Remove numbers
- **stripWhitespace()**: Remove excess whitespace
- **bracketX()**: Remove all text within brackets (e.g. "It's (so) cool" becomes "It's cool")
- **replace_abbreviation()**: Replace abbreviations with their full text equivalents (e.g. "Sr" becomes "Senior")
- **replace_contraction():** Convert contractions back to their base words (e.g. "shouldn't" becomes "should not")
- **replace_symbol()** Replace common symbols with their word equivalents (e.g. "$" becomes "dollar")
- **removeWords,stopwords('english'):**replaces stop words from text data (e.g. a,an,about,again etc)

Creating corpus words:

Let's build a corpus out of this vector of strings. A corpus is a collection of documents,

There are two kinds of the corpus data type, the permanent corpus, i.e. PCorpus, and the volatile corpus, i.e. VCorpus.

The difference between the two has to do with how the collection of documents is stored in your computer. We will use the volatile corpus, which is held in computer's RAM rather than saved to disk, just to be more memory efficient.

And the tm package provides what are called Source functions to do just that! In this exercise, we'll use a Source function called VectorSource() because our text data is contained in a vector.

The output of this function is called a Source object.

Creating document term matrix

A document-term matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents.

In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

After setting sparsity threshold to 0.97 ,number of documents have been reduced to 99

Sparsity:percentage of zeros in columns

Density:percentage filled with non zeroes

Sparsity +density=100%

```
> corpus=removeSparseTerms(corpus, .97)

> inspect(corpus)

<<DocumentTermMatrix (documents: 312735, terms: 99)>>

Non-/sparse entries: 1709612/29251153

Sparsity           : 94%

Maximal term length: 11

Weighting          : term frequency - inverse document frequency (tf-idf)
```

Chapter 4:

Modeling

Model Selection

From the data there are 6 target variables and one predictor variable

So it comes under multi label classification

These types of problems, where we have a set of target variables, are known as **multi-label classification** problems

If there are multiple categories but each instance is assigned only one, therefore such problems are known as **multi-class classification** problem.

If there are multiple labels as target variables then it is called multi label classification

This method can be carried out in three different ways as:

1. Binary Relevance
2. Classifier Chains
3. Label Powerset

Binary relevance

In binary relevance, this problem is broken into different single class classification problems and model is applied to each target variable

Classifier Chains

In this, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain.

Label Powerset

In this, we transform the problem into a multi-class problem with one multi-class classifier is trained on all unique label combinations found in the training data.

In the model I am using binary equivalence method

As from the experiments binary relevance method has given best accuracy rate than compared to other methods ,and also from sources like internet and books I got to know that binary relevance method is the best method for building a multi label model.

I have used random forest algorithm in caret train function

Random forest algorithm :

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees.

We can do classification and regression using random forest

In general, the **more trees in the forest** the more robust the forest looks like. In the same way in the random forest classifier, the **higher the number** of trees in the forest gives **the high accuracy** results

Random Forest pseudocode:

1. Randomly select **k** features from total **m** features.
    1. Where **k << m**
2. Among the **k** features, calculate the node **d** using the best split point.
3. Split the node into **daughter nodes** using the **best split**.
4. Repeat **1 to 3** steps until l number of nodes has been reached.
5. Build forest by repeating steps **1 to 4** for n number times to create **n number of trees**.

The beginning of random forest algorithm starts with randomly selecting **k** features out of total **m** features. In the image, you can observe that we are randomly taking features and observations.

Random forest prediction pseudocode:

To perform prediction using the trained random forest algorithm uses the below pseudocode.

1. Takes the **test features** and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the **votes** for each predicted target.
3. Consider the **high voted** predicted target as the **final prediction** from the random forest algorithm.

To perform the prediction using the trained random forest algorithm we need to pass the test features through the rules of each randomly created trees. Suppose let’s say we formed 100 random decision trees to from the random forest.

Each random forest will predict different target (outcome) for the same test feature. Then by considering each predicted target votes will be calculated.

Suppose the 100 random decision trees are prediction some 3 unique targets **x, y, z** then the votes of x is nothing but out of 100 random decision tree how many trees prediction is **x.**

Likewise for other 2 targets (y, z). If x is getting high votes. Let's say out of 100 random decision tree **60** trees are predicting the target will be x. Then the final random forest returns the x as the predicted target. This concept of voting is known as **majority voting**.

Below are some the application where random forest algorithm is widely used.

1. Banking
2. Medicine
3. Stock Market
4. E-commerce

Modelling results

```
+ Fold01: mtry= 2
- Fold01: mtry= 2
…………..
+ Fold10: mtry=99
- Fold10: mtry=99
Aggregating results
Selecting tuning parameters
Fitting mtry = 2 on full training set
Random Forest
  99 predictor
   2 classes: '0', '1'
Resampling: Cross-Validated (10 fold)
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7525235  0.4000533
  50    0.7439312  0.3668774
  99    0.7412814  0.3648317

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was mtry = 2.
```

Toxic

```
+ Fold01: mtry= 2
- Fold01: mtry= 2
+ Fold01: mtry=50
+ Fold10: mtry=99
- Fold10: mtry=99
```

```
Aggregating results
Selecting tuning parameters
Fitting mtry = 50 on full training set
Random Forest
  99 predictor
   2 classes: '0', '1'

Resampling: Cross-Validated (10 fold)
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8780814  0.7541204
  50    0.9005782  0.7996186
  99    0.8947857  0.7876835

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was mtry = 50.
```

Insult

```
+ Fold01: mtry= 2
- Fold01: mtry= 2
……...
+ Fold10: mtry=99
- Fold10: mtry=99
Aggregating results
Selecting tuning parameters
Fitting mtry = 50 on full training set
Random Forest
  99 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7326698  0.2953704
  50    0.7369772  0.3298983
  99    0.7275412  0.3087186

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was mtry = 50.
```

severe_toxic

```
+ Fold01: mtry= 2
```

```
- Fold01: mtry= 2
+ Fold10: mtry=99
- Fold10: mtry=99
Aggregating results
Selecting tuning parameters
Fitting mtry = 2 on full training set
Random Forest

  99 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8719599  0.002850718
  50    0.8468152  0.171133504
  99    0.8436684  0.174554855

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was mtry = 2.
```

identity_hate

```
+ Fold01: mtry= 2
- Fold01: mtry= 2
+ Fold10: mtry=99
- Fold10: mtry=99
Aggregating results
Selecting tuning parameters
Fitting mtry = 2 on full training set
Random Forest
  99 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8535988  0.008812626
  50    0.8363972  0.238206610
  99    0.8357385  0.240691465

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was mtry = 2.
```

Threat

```
+ Fold01: mtry= 2
- Fold01: mtry= 2
……..
+ Fold10: mtry=99
- Fold10: mtry=99
Aggregating results
Selecting tuning parameters
Fitting mtry = 2 on full training set
Random Forest

  99 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8279548  0.05018278
  50    0.8200181  0.29498085
  99    0.8155553  0.28317024

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was mtry = 2.
```

Model evaluation

Confusion matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm.

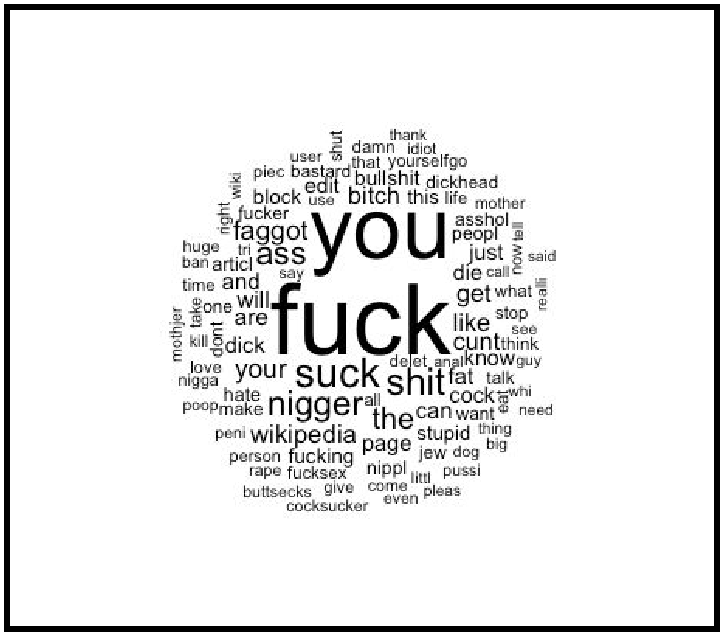
Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

As we don’t have real test data to test the prediction accuracy we are not testing the model.we are only determining the accuracy of the model

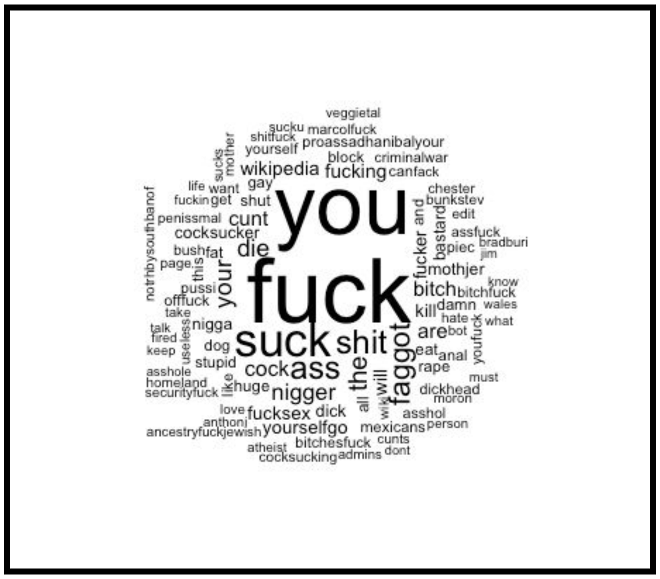
Extra figures:

Word cloud of comments :

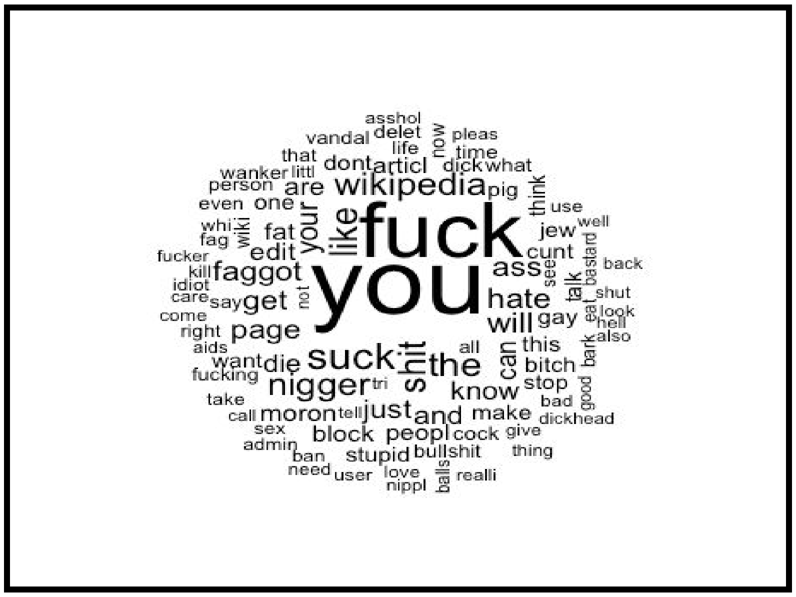
1.which are obscene



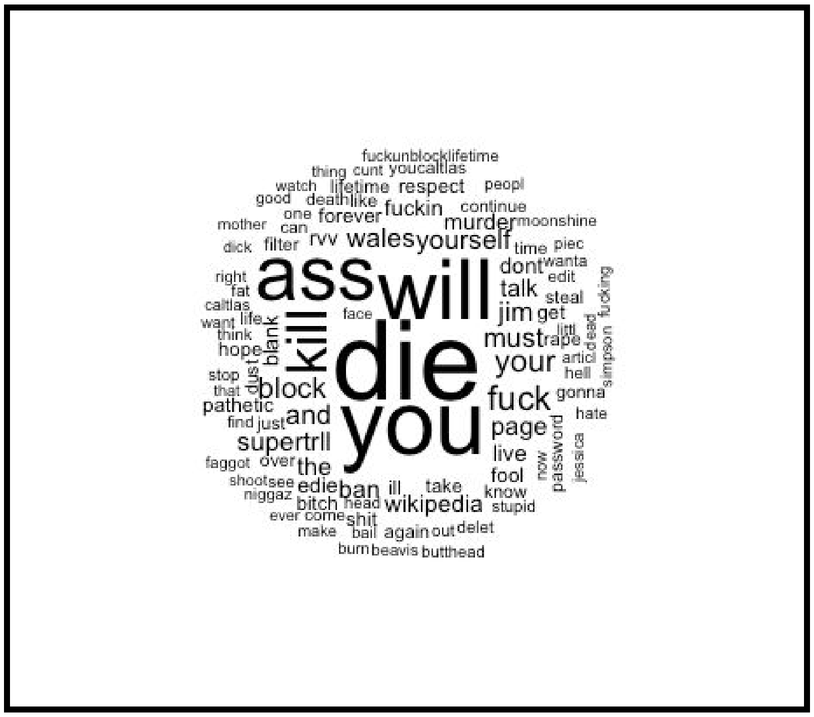
2.Severe_toxic



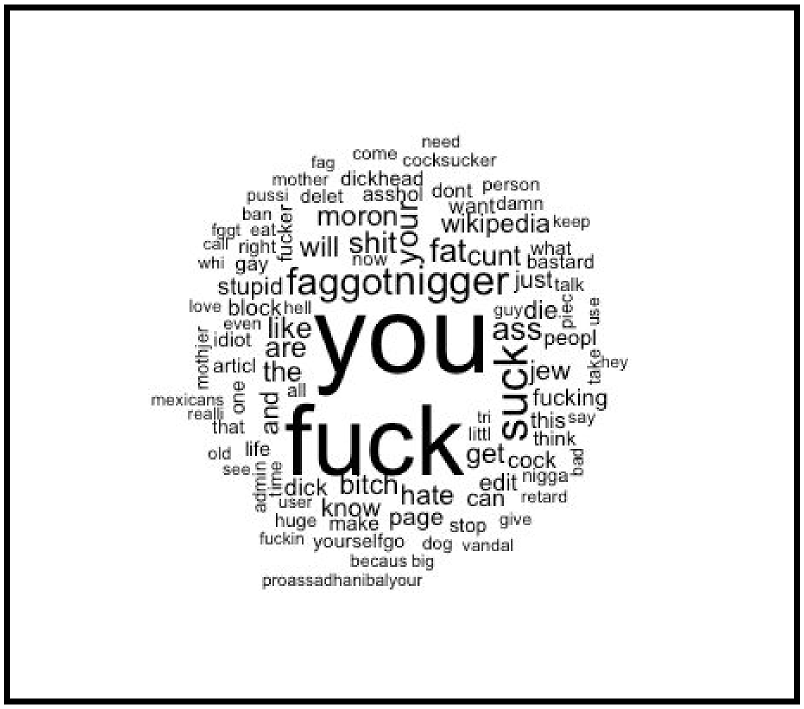
3.Toxic



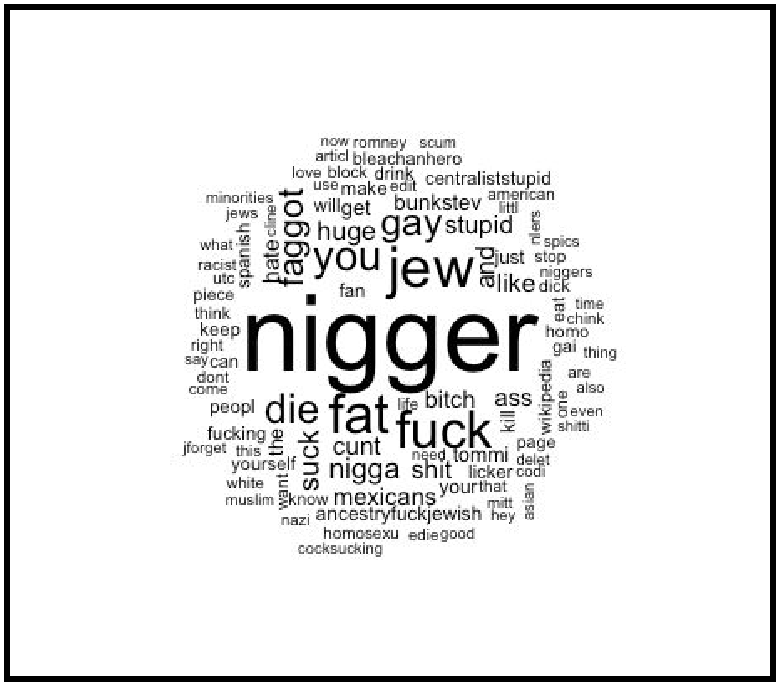
4.Threat



5.Insult



6.Identity_hate



Complete R code

with comments explaining the steps:

<u>**Warning:**</u>

<u>**As data is huge it took some hours for me to execute the process.So make sure your computer is fast enough(RAM) to execute the process**</u>

```
#reading data

train_data=read.csv('train.csv',na.strings=c("",NA),stringsAsFactors
= F,sep=',')

test_data=read.csv('test.csv',na.strings=c("",NA))

submission=read.csv('sample_submission.csv')



#empty values

sapply(train,function(x) sum(is.na(x)))

sapply(test,function(x) sum(is.na(x)))

targets =
c('obscene','insult','toxic','severe_toxic','identity_hate','threat')

#unlabelled(clean) comments

unlabelled_comments = sum((train_data$toxic!=1) &
(train_data$severe_toxic!=1) & (train_data$obscene!=1) &

                              (train_data$threat!=1) &
(train_data$insult!=1) & (train_data$identity_hate!=1))

#unlabelled_comments percentage
```

```
print(unlabelled_comments/159571)

for (category in
c('obscene','insult','toxic','severe_toxic','identity_hate','threat')
){

  print(category)

  print('percentage of ones')

  print(sum(train_data[category]==1)*100/159571)

  print('percentage of zeroes')

  print(sum(train_data[category]==0)*100/159571)

  print('number of empty cells')

  print(sum(train_data[category]==""))

}

library(corrgram)

corrgram(cor(train1[,3:8]),order=TRUE,main="plot",upper.panel=panel.c
or,text.panel=panel.txt)

#count of words in each comment

library(stringr)

for (i in (1:length(train_data))){

  train_data[i,9]=str_count(train_data[i,2])

}

#plotting histogram of words

hist(train_data[,9],main = 'Histogram of words',xlab = 'number of
words in comments')
```

```
#Clean up the comment text

find    <- c("what's","\'s","\'ve","can't","n't","i'm","\'re",

             "\'d","\'ll","\'scuse","\'W",'\'s+',"\"i",

             "\"\"","\n\n","there's",'<','else' )

replace <- c("what is "," "," have ","cannot " ," not ","i am ",

             " are "," would "," will "," excuse ",' ',' ',"i","","",

             "there is","","")
```

#replacing some words

```
library(qdap)

mgsub(find,key,train_data[,2])

mgsub(find,key,test_data[,2])

library(tm)
```

#combining train and test data

```
combined_data=rbind(train_data[,1:2],test_data)
```

#creating source

```
corpus=Corpus(VectorSource(combined_data[,2]))
```

#Remove all punctuation marks

```
corpus=tm_map(corpus,removePunctuation)
```

#Make all characters lowercase

```
corpus=tm_map(corpus,tolower)
```

#Remove text within brackets

```
corpus=tm_map(corpus,bracketX)

#Replace abbreviations

corpus=tm_map(corpus,replace_abbreviation)

#Replace contractions

corpus=tm_map(corpus,replace_contraction)

#Replace symbols with words

corpus=tm_map(corpus,replace_symbol)

#removing words

corpus=tm_map(corpus,removeNumbers)

#removing stop words

corpus=tm_map(corpus,removeWords,stopwords('english'))

#removing white space

corpus=tm_map(corpus,stripWhitespace)

#creating a corpus of words as document term matrix

corpus=DocumentTermMatrix(corpus,control = list(weighting =
function(x) weightTfIdf(x, normalize = FALSE)))

#sparsity threshold

corpus=removeSparseTerms(corpus, .97)

#inspecting corpus

inspect(corpus)

#converting to data frame

dat=as.data.frame(unlist(as.matrix(corpus)))
```

```
#converting to factors

for (i in (1:length(dat))){

  dat[,i]=factor(dat[,i],labels=c(1:length(unique(dat[,i]))))

}

dat1=factor(dat1,levels=c(1:length(unique(dat1))))


#converting to factors

for (i in (3:8)){

  train_data[,i]=factor(train_data[,i])

}

#saperating train corpus


train_corpus=dat[1:159571,]

dat1=dat[-(1:159571),]


#combinng text rows back to test data

test_corpus=dat
```

```
#model

library(caret)

#random forest using K-fold and down sampling with

#binary relevance method

for (category in
c('obscene','insult','toxic','severe_toxic','identity_hate','threat')
){

  training_data=cbind(train_data[category],train_corpus)

  listoffactors=c(colnames(train_corpus))

newGrid = expand.grid(mtry = c(2,50,99))



  down_sample=downSample(x=training_data[,2:100],y=training_data[,1]

                        ,yname = category)

  train_control= trainControl(method="cv", number=5)

  model= train(reformulate(termlabels = listoffactors, response =
category),data=down_sample,trControl=train_control,method="rf",tuneGri
d = newGrid

,family=binomial('logit'))



    pred=predict(model,test_corpus,type='raw')

     print(category)

     summary(model)

#storing predict results of a category to submission file
```

```
     submission[category]=pred
```

#removing that category as we are doing binary relevance

```
     training_data[category]=NULL



}
```

#using classifier chain method

```
for (category in
c('obscene','insult','toxic','severe_toxic','identity_hate','threat')
){

  training_data=cbind(train_data[category],train_corpus)

  listoffactors=c(colnames(train_corpus))

down_sample=downSample(x=training_data[,2:100],y=training_data[,1]

                         ,yname = category)

  train_control= trainControl(method="cv", number=5)

newGrid = expand.grid(mtry = c(2,50,99))

listoffactors=reformulate()

 model_classchain= train(reformulate(termlabels = listoffactors,
response=category),data=down_sample,trControl=train_control,

method="rf",ntree=100,tuneGrid = newGrid,family=binomial('logit'))



  pred=predict(model_classchain,test_corpus,type='raw')

  print(category)

  summary(model)
  #storing predict results of a category to submission file

  submission[category]=pred



}

#wordcloud for different categories

library(SnowballC)

library(wordcloud)

myfunction=function(i) {

  print('word cloud of')

  print(colnames(train_data)[i])

  type=as.data.frame(train_data[train_data[,i]==1,])

  corpus1=Corpus(VectorSource(type[,2]))

  corpus1=tm_map(corpus1,removeWords,stopwords('english'))

  corpus1=tm_map(corpus1,removeNumbers)

  corpus1=tm_map(corpus1,removePunctuation)

  corpus1=tm_map(corpus1,stemDocument)

  wordcloud(corpus1, max.words = 100, random.order = FALSE)

}

#i can take values from 3 to 8

 myfunction(i)
```

References:

Machine learning with R by Bretty lantz

Text Mining with R: A Tidy Approach

Towards data science