Misinformation Analysis Report

Cynthia Lam

8/10/2020

Contents

1. Introduction

Misinformation is any inaccurate information, while disinformation refers to information fabricated deliberatively intended to deceive. In a time like this, the complicated political situation and the COVID-19 pandemic, together with the rapid propagation of information on social media, have enabled a rapid increase in the amount of information, which is often referred to as "an infodemic" (Zarocostas, 2020). This overabundance of information consists of both accurate and inaccurate information (misinformation and disinformation), making it challenging for the general public to distinguish facts from fake news.

This project aims to explore a dataset of news articles published in the US (Bisaillon, 2020) and develop a model based on the text mined from the dataset to predict news accuracy. The report outlines the methodology of data exploration and wrangling, as well as building a model to predict information accuracy/validity using Naive Bayes, LDA, LQA, GLM, KNN and Random Forest. To evaluate the performance of the models, the accuracy of prediction will be performed on the test set across all models.

2. Methods

2.1 Preprocessing - Data Preparation

```
url <- "https://github.com/thialam/Misinformation-Analysis/raw/master/True.csv"
True <- read_csv(url)
url <- "https://github.com/thialam/Misinformation-Analysis/raw/master/Fake.csv"
Fake <- read_csv(url)
True <- True %>% mutate(validity = 1) #adding a column for validity where 1 = true
Fake <- Fake %>% mutate(validity = 0) #where 0 = fake
news <- full_join(True,Fake) #combining both dataframes
news <- news %>% filter(!is.na(text))
set.seed(1, sample.kind = "Rounding")
```

```r
news <- sample_n(news,20000) #sampling 20000 from the dataset to accommodate the limited computational p
news$validity <- as.factor(news$validity)
news$subject <- as.factor(news$subject)
```

To ease data downloading to R, I have downloaded the dataset from Kaggle and re-uploaded it to my Github, but the dataset is the same as the original one from Kaggle. The original dataset is >40K long and each data entry (row) contains a lot of textual data (it is the entire article in one row), we will only sample 20K entries from the dataset after removing rows with NAs.

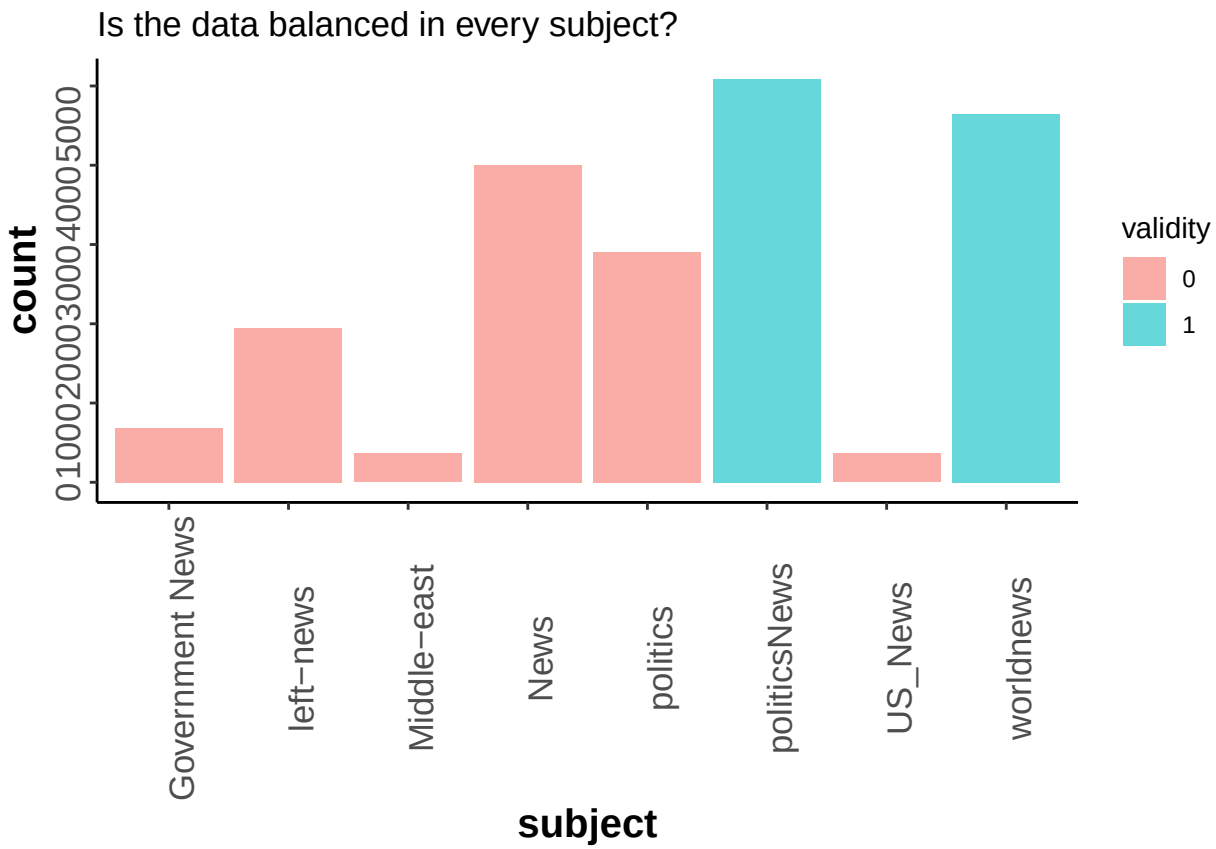
2.2 Preprocessing - Data Exploration and Wrangling

In the process of exploring the data through visualisations, a few problems about the dataset have been identified. Hence, we will also perform data cleaning and wrangling in this section. #### 2.2.1 Inspecting the given data and basic cleaning Let's take a look at the first few rows of the original dataset.

```
## # A tibble: 6 x 5
##   title                     text                      subject  date     validity
##   <chr>                     <chr>                     <fct>    <chr>    <fct>
## 1 U.S. creating 'sensation~ BEIJING (Reuters) - The ~ worldne~ Decembe~ 1
## 2 Brazil's Congress reject~ BRASILIA (Reuters) - Bra~ worldne~ October~ 1
## 3 BUSTED: Trump Supporter ~ Clearly, there is no low~ News     Novembe~ 0
## 4 HERE'S WHY L.L. BEAN Is ~ President-elect Donald T~ left-ne~ Jan 12,~ 0
## 5 Facebook's political inf~ (Reuters) - As the U.S. ~ politic~ June 29~ 1
## 6 NEW VIDEO Of United Airl~ Watch here:@United overb~ left-ne~ Apr 12,~ 0
```

We can see it consists of 5 variables - title, text, date, subject (category) and validity (whether it is true or fake). First, we inspect whether the dataset is balanced in terms of validity:

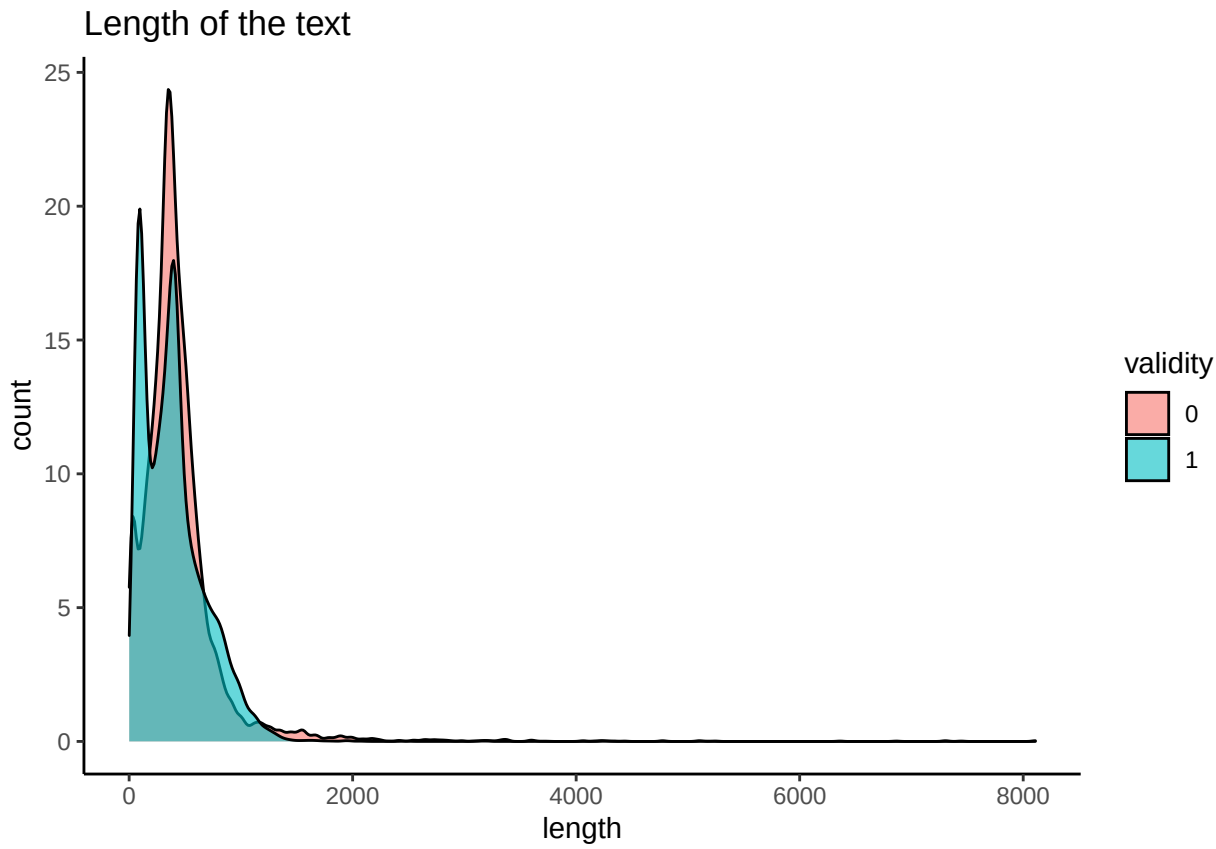
Then, we look at whether the dataset is balanced in terms of subjects (news categories).



We can see that the data is skewed - the subjects that are present in "true" set are not present in the "fake" set. In other words, we need to remove subjects from the dataset as using it as correlating factors produce unreasonably accurate results.

```
news<-news[-3] #turns out the subjects are completely different in the categories, so we need to delete
```

Let's also inspect the distribution of the lengths of the text



We can see that the distributions overlap, indicating the length does not seem to be a good predictor.

We will also combine the title and text column and give each row a unique ID for further wrangling later.

```
news <- news %>% # Combine 'title' & 'text' column
  unite(col = text ,title, text, sep = ' ')  %>%
  mutate(ID = as.character(1:nrow(news))) #add a unique ID for each row
```

2.2.2 Text mining

Articles are very content-rich, but it also constitutes many components that are not informative in this case, such as numbers, punctuations, stopwords (e.g. and, is, a an, etc.) To perform text mining, "tm" (Feinerer, 2019) and "textstem" (Rinker, 2018) packages are required.

```
text <- VCorpus(VectorSource(news$text))  #create a corpus for tm for text mining and cleaning
writeLines(as.character(text[[1]]))
```

```
## U.S. creating 'sensational hype' over China's military modernization: ministry BEIJING (Reuters) - T
```

```
text <- tm_map(text, content_transformer(tolower))
text <- tm_map(text, removeNumbers)
text <- tm_map(text, content_transformer(str_remove_all),
    "[[:punct:]]")
text <- tm_map(text, removeWords, stopwords("english"))
text <- tm_map(text, stripWhitespace)

paragraph1 <- as.character(text[[1]])  #inspect to see if all the uninformative words and punctuations
print(paragraph1)
```

```
## [1] "us creating sensational hype chinas military modernization ministry beijing reuters united stat
```

Here, we can see the differences made after removing the uninformative components, but there is more we can do to further condense the information, by utilising lemmatisation. Lemmatisation refers to removing inflectional endings to return the base or dictionary form of a word (known as the lemma) by morphological analysis of words (Manning and Schutze, 1999).

```
text <- tm_map(text, content_transformer(lemmatize_strings))  #remove  strings to return only the meani
paragraph2 <- as.character(text[[1]])  #inspect to see if inflectional endings have been removed and on
print(paragraph2)
```

```
## [1] "us create sensational hype china military modernization ministry beijing reuters unite state cr
```

Here, we can see the returned text only contains the lemma - we are now ready to utilise the cleaned data.

Next, we create document term matrix to inspect the popularity of words and also transition into a clean dataframe.

```
dtm <- DocumentTermMatrix(text) #create document term matrix to view the popularity of words and also t
dtm <- removeSparseTerms(dtm, sparse = 0.99) #remove sparse terms
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 20000, terms: 2512)>>
## Non-/sparse entries: 2367154/47872846
## Sparsity           : 95%
## Maximal term length: 16
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
```

```
## Docs    make one people president republican say state trump will year
##   10695    7  14      2         4          0   5    10     0    4    5
##   11416    6   8      6         9          6   1    46     5    4    3
##   12588    5  19      3        13          4  18    13    13    4    3
##   2663     4  18      4         2          0  25    15     0   10    9
##   411      3  24     13        12          9   8    80     1   10    6
##   4466     5  19      3        13          4  18    13    13    4    3
##   4974     4  18      4         2          0  25    15     0   10    9
##   5280     5  11     13         0          0  19    16     1    2   11
##   681      6   7      6         8          5   1    45     1    3    2
##   9163     6   8      6         9          6   1    46     5    4    3
```

```
df <- tidy(dtm)
```

What are the top terms?

```
## # A tibble: 2,512 x 2
##    term        freq
##    <chr>      <dbl>
##  1 say        77456
##  2 trump      63885
##  3 state      27995
##  4 will       25259
##  5 president  24521
##  6 people     18821
##  7 republican 17977
##  8 make       17565
##  9 one        16852
## 10 year       15691
## # ... with 2,502 more rows
```

What about the top terms for fake and real news respectively?

```
realwords <- df %>% inner_join(news, by = c('document' = 'ID')) %>%
  select(-document) %>%
  filter(validity==1) %>%
  group_by(term) %>%
  summarize(real_freq = sum(count)) %>%
  arrange(desc(real_freq))
realwords
```

```
## # A tibble: 2,508 x 2
##    term       real_freq
##    <chr>          <dbl>
##  1 say            53751
##  2 trump          27170
##  3 state          17221
##  4 reuters        12902
##  5 president      12357
##  6 will           12159
##  7 republican     10514
##  8 year            8991
##  9 government      8957
## 10 house           8562
## # ... with 2,498 more rows
```

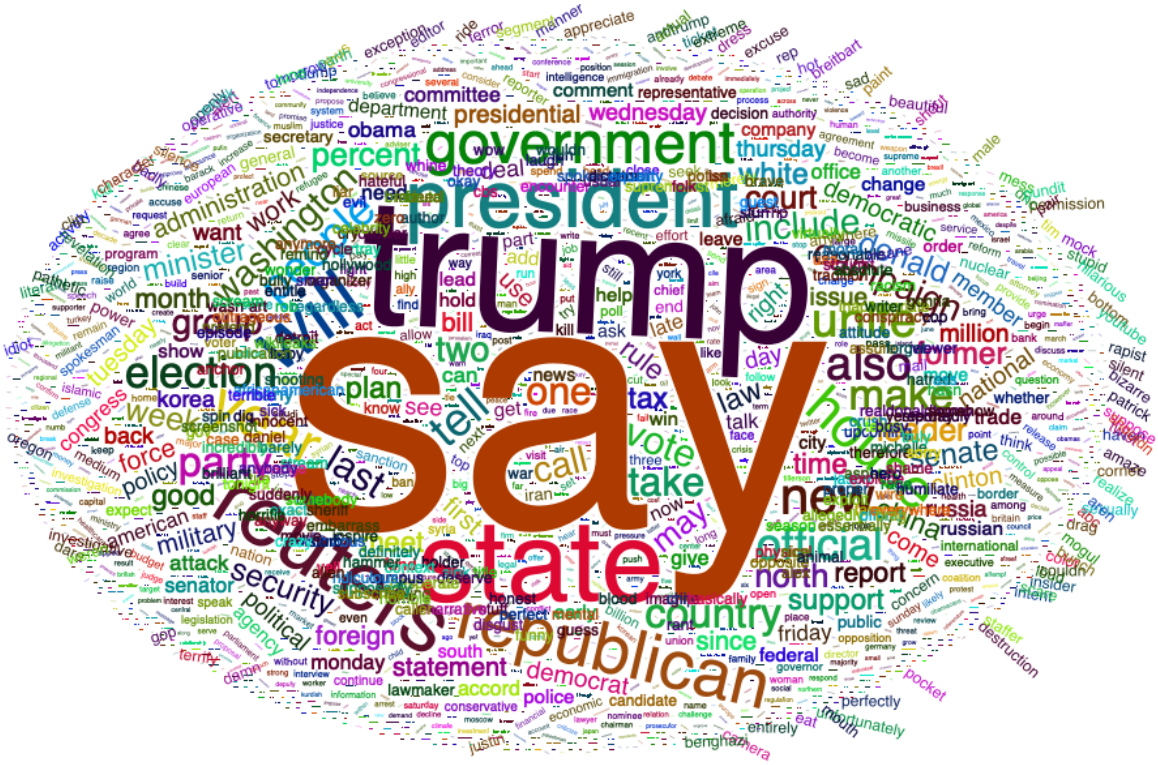
```
fakewords <- df %>% inner_join(news, by = c('document' = 'ID')) %>% #top terms for fake news
    select(-document) %>%
```

```
    filter(validity==0) %>%
    group_by(term) %>%
    summarize(fake_freq = sum(count)) %>%
    arrange(desc(fake_freq))
fakewords
```

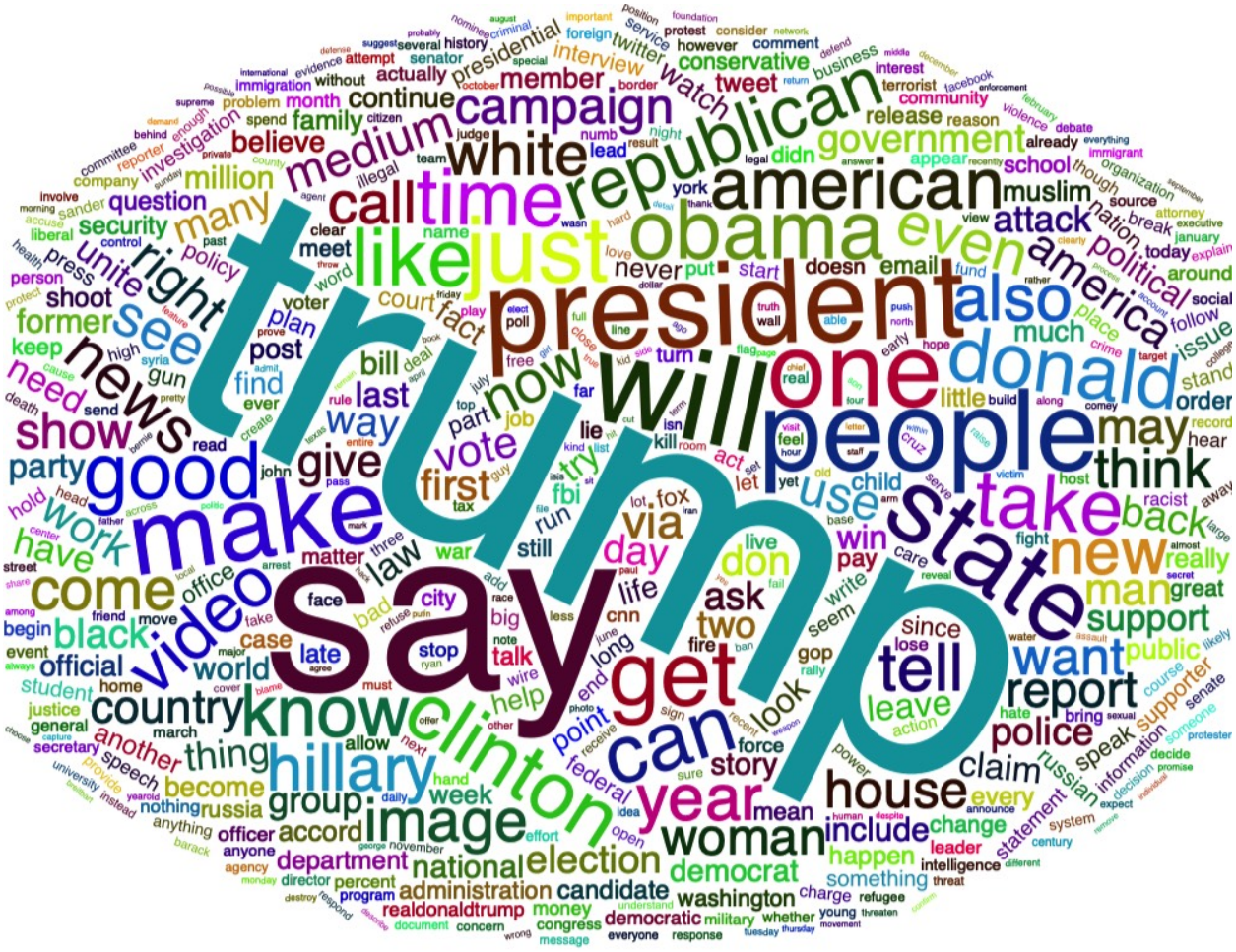
```
## # A tibble: 2,512 x 2
##    term      fake_freq
##    <chr>         <dbl>
##  1 trump         36715
##  2 say           23705
##  3 will          13100
##  4 president     12164
##  5 people        11780
##  6 make          11003
##  7 one           10805
##  8 state         10774
##  9 get            9565
## 10 just           9536
## # ... with 2,502 more rows
```

Let's visualise the frequency of the terms better using word cloud.

For real news:



For fake news:



We can see that the popularity of different terms varries across the two datasets. Next, we combine the two lists with the frequencies for both fake and real news and calculate the ratio of fake frequency/total frequency.

```
termslist <- fakewords %>% full_join(realwords, by = c('term'='term')) #combine the two lists
termslist <- termslist %>% mutate (ratio = fake_freq/(real_freq+fake_freq)) #find out the ratio for fak
```

There are too many terms and the computer will not be able to manage, so we are only filtering to only keep the terms with frequency higher than 1500 in either category, as well as with a ratio of larger than 0.65 or smaller than 0.35. Please note this is not ideal and the thresholds set for the frequency and ratio are arbitrary - if computational power allows, please try to perform the training using the full terms list.

```
sectermslist <- termslist %>%
  arrange(desc(ratio)) %>%
  filter(!(ratio <=0.65 & ratio >=0.35)) %>%
  filter(!(fake_freq<=1500 & real_freq <=1500)) %>%
  filter(!grepl('reuters', term)) #removing "Reuters" as it makes the data too biased - with a ratio of
```

The term “reuters” is also removed as with a ratio of 0.015, it makes the data highly biased. In real life scenario, not that many real news articles contain the word “reuters”. If we included it in the dataset, it would have caused overfitting with this current dataset.

2.3 Dataset cleaning and train/test preparation

Now we have a dataframe of segregated, informative terms - we need to combine it back with the news dataframe with validity before partitioning our data into training and testing sets. We can see now each row contains information about the article, its validity and the frequencies of different important terms we have identified through text mining in the article.

An 80/20 train/test split has been adopted based on the Pareto Principle (80% of effects come from 20% of causes (Bunkley,2008))

```
news$validity <- as.factor(news$validity)
news1 <- df %>% #combining the dataframe of segregated informative terms back with the news frame
  filter(df$term %in% sectermslist$term) %>% #only selecting the selected terms we filtered out
  inner_join(news, by = c('document' = 'ID')) %>%
  spread(term,count) %>% #reshape the data so each document takes up one row
  mutate_all(~ifelse(is.na(.), 0, .)) %>%  #turn NA into 0
  select(-c(document, date, text)) #removing document ID, date and the full text

head(news1)
```

```
## # A tibble: 6 x 168
##   validity actually agency agreement  ally america american anyone anything
##      <int>    <dbl>  <dbl>     <dbl> <dbl>   <dbl>    <dbl>  <dbl>    <dbl>
## 1        2        0      0         0     0       1        0      0        0
## 2        2        0      0         0     1       1        0      0        0
## 3        1        0      0         0     0       0        0      0        0
## 4        1        0      0         0     0       1        0      0        0
## 5        2        0      0         0     0       0        0      0        1
## 6        1        0      0         0     0       0        0      0        0
## # ... with 159 more variables: area <dbl>, authority <dbl>, away <dbl>,
## #   bad <dbl>, bank <dbl>, believe <dbl>, billion <dbl>, black <dbl>,
## #   'break' <dbl>, britain <dbl>, budget <dbl>, can <dbl>, care <dbl>,
## #   chief <dbl>, child <dbl>, china <dbl>, claim <dbl>, clinton <dbl>,
## #   cnn <dbl>, coalition <dbl>, committee <dbl>, community <dbl>,
## #   congress <dbl>, course <dbl>, cut <dbl>, deal <dbl>, deny <dbl>,
## #   didn <dbl>, doesn <dbl>, don <dbl>, early <dbl>, economic <dbl>,
## #   email <dbl>, european <dbl>, even <dbl>, event <dbl>, ever <dbl>,
## #   every <dbl>, expect <dbl>, fact <dbl>, fbi <dbl>, feel <dbl>,
## #   foreign <dbl>, fox <dbl>, friday <dbl>, get <dbl>, good <dbl>, gop <dbl>,
## #   government <dbl>, governor <dbl>, great <dbl>, gun <dbl>, hand <dbl>,
## #   happen <dbl>, hate <dbl>, have <dbl>, hillary <dbl>, history <dbl>,
## #   host <dbl>, illegal <dbl>, image <dbl>, international <dbl>, iran <dbl>,
## #   islamic <dbl>, just <dbl>, know <dbl>, korea <dbl>, lawmaker <dbl>,
## #   leader <dbl>, legislation <dbl>, let <dbl>, liberal <dbl>, lie <dbl>,
## #   life <dbl>, like <dbl>, live <dbl>, look <dbl>, lot <dbl>, man <dbl>,
## #   matter <dbl>, mean <dbl>, measure <dbl>, medium <dbl>, military <dbl>,
## #   minister <dbl>, missile <dbl>, monday <dbl>, month <dbl>, much <dbl>,
## #   muslim <dbl>, never <dbl>, news <dbl>, night <dbl>, north <dbl>,
## #   nothing <dbl>, now <dbl>, nuclear <dbl>, obama <dbl>, officer <dbl>,
## #   official <dbl>, ...
```

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = news1$validity, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- news1[-test_index,]
test_set <- news1[test_index,]
train_set$validity <- as.factor(train_set$validity)
test_set$validity <- as.factor(test_set$validity)
table(train_set$validity) #is it balanced?
```

```
##
##    1    2
## 8245 7748
```

```
table(test_set$validity)
```

```
##
##    1    2
## 2011 1988
```

We finished preprocessing of the data - now we are ready to work on building our models.

2.4 Models training

We will be utilising Naive Bayes, LDA, QDA, GLM, KNN and Random Forest to build different models for prediction.

2.4.1 Naive Bayes

First, we utilised Naive Bayes, which is a kind of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features (McCallum, 2019).

We will inspect the confusion matrix for each model, but the comparison will be made based on only the overall accuracy. A table is built to store the accuracy for comparison later.

```
train_nb <- naive_bayes(validity ~ ., data = train_set)
y_hat0 <- predict(train_nb, newdata = test_set)
confusionMatrix(data = y_hat0, reference = test_set$validity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 1637  183
##          2  374 1805
##
##                Accuracy : 0.8607
##                  95% CI : (0.8496, 0.8713)
##     No Information Rate : 0.5029
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7216
##
##  Mcnemar's Test P-Value : 8.242e-16
##
##             Sensitivity : 0.8140
##             Specificity : 0.9079
##          Pos Pred Value : 0.8995
##          Neg Pred Value : 0.8284
##              Prevalence : 0.5029
##          Detection Rate : 0.4094
##    Detection Prevalence : 0.4551
##       Balanced Accuracy : 0.8610
##
##        'Positive' Class : 1
##
```

```r
mean0 <- mean(y_hat0 == test_set$validity)
results <- data_frame(Method = "Naive Bayes", Accuracy = mean0)
results %>% knitr::kable()
```

Method	Accuracy
Naive Bayes	0.8607152

2.4.2 Linear Discriminant Analysis

Next, we utilise Linear Discriminant Analysis (LDA). LDA reduces dimentionality of the data by projecting the input to a linear subspace. (Sckit Learn, 2020)

```r
set.seed(1, sample.kind = "Rounding")
train_lda <- train(validity ~ ., method = "lda", data = train_set)
y_hat1 <- predict(train_lda, test_set)
mean1 <- mean(y_hat1 == test_set$validity)
confusionMatrix(data = y_hat1, reference = test_set$validity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 1871   49
##          2  140 1939
##
##                Accuracy : 0.9527
##                  95% CI : (0.9457, 0.9591)
##     No Information Rate : 0.5029
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9055
##
##  Mcnemar's Test P-Value : 5.889e-11
##
##             Sensitivity : 0.9304
##             Specificity : 0.9754
##          Pos Pred Value : 0.9745
##          Neg Pred Value : 0.9327
##              Prevalence : 0.5029
##          Detection Rate : 0.4679
##    Detection Prevalence : 0.4801
##       Balanced Accuracy : 0.9529
##
##        'Positive' Class : 1
##
```

```r
results <- bind_rows(results,
                     data_frame(Method="LDA",
                                Accuracy = mean1 ))
results %>% knitr::kable()
```

Method	Accuracy
Naive Bayes	0.8607152
LDA	0.9527382

2.4.3 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is a sister to LDA, where it also acts in the same vein but now with a quadratic decision surface.

```
set.seed(1, sample.kind = "Rounding")
train_qda <- train(validity ~ ., method = "qda", data = train_set)
y_hat2 <- predict(train_qda, test_set)
mean2 <- mean(y_hat2 == test_set$validity)
confusionMatrix(data = y_hat2, reference = test_set$validity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 1701  173
##          2  310 1815
##
##                Accuracy : 0.8792
##                  95% CI : (0.8687, 0.8892)
##     No Information Rate : 0.5029
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7585
##
##  Mcnemar's Test P-Value : 6.085e-10
##
##             Sensitivity : 0.8458
##             Specificity : 0.9130
##          Pos Pred Value : 0.9077
##          Neg Pred Value : 0.8541
##              Prevalence : 0.5029
##          Detection Rate : 0.4254
##    Detection Prevalence : 0.4686
##       Balanced Accuracy : 0.8794
##
##        'Positive' Class : 1
##
```

```
results <- bind_rows(results,
                     data_frame(Method="QDA",
                                Accuracy = mean2 ))
results %>% knitr::kable()
```

Method	Accuracy
Naive Bayes	0.8607152
LDA	0.9527382
QDA	0.8792198

We can see it outperforms LDA quite significantly.

2.4.4 Generalized linear model

In general linear model (GLM), general refers to predicting using more than one explanatory variable (v.s. the simple linear model). It is made up of a linear predictor, a link function and a variance function.

```r
set.seed(1, sample.kind = "Rounding")
train_glm <- train(validity ~ ., method = "glm", data = train_set)
y_hat3 <- predict(train_glm, test_set)
mean3 <- mean(y_hat3 == test_set$validity)
confusionMatrix(data = y_hat3, reference = test_set$validity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 1930   65
##          2   81 1923
##
##                Accuracy : 0.9635
##                  95% CI : (0.9572, 0.9691)
##     No Information Rate : 0.5029
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.927
##
##  Mcnemar's Test P-Value : 0.2145
##
##             Sensitivity : 0.9597
##             Specificity : 0.9673
##          Pos Pred Value : 0.9674
##          Neg Pred Value : 0.9596
##              Prevalence : 0.5029
##          Detection Rate : 0.4826
##    Detection Prevalence : 0.4989
##       Balanced Accuracy : 0.9635
##
##        'Positive' Class : 1
##
```

```r
results <- bind_rows(results,
                     data_frame(Method="GLM",
                                Accuracy = mean3 ))
results %>% knitr::kable()
```

Method	Accuracy
Naive Bayes	0.8607152
LDA	0.9527382
QDA	0.8792198
GLM	0.9634909

```r
varImp(train_glm)
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 167)
##
##            Overall
## say         100.00
## via          61.12
## video        58.26
## wire         45.81
```

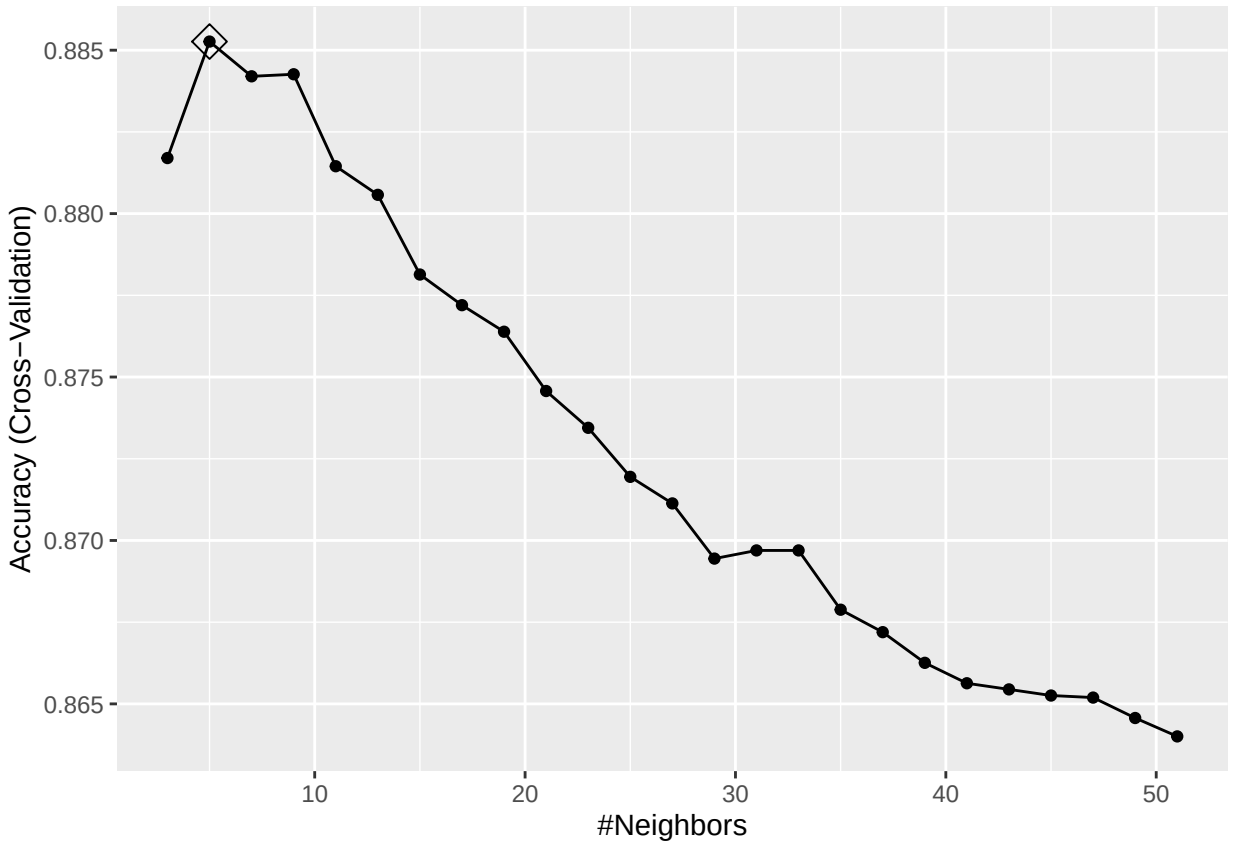
```
## have          33.73
## obama         33.55
## doesn         33.15
## wednesday     31.56
## don           30.82
## press         30.18
## tuesday       29.36
## just          29.33
## didn          29.22
## friday        28.79
## washington    28.66
## monday        28.54
## liberal       27.17
## thursday      25.50
## gop           24.74
## american      24.18
```

It outperforms LDA or QDA.

2.4.5 K Nearest Neighbours

K-nearest neighbours is a function where we approximate the function locally based on the neighbouring objects. However, what is the optimal number for K that we should use? We will also do one with k-fold cross-validation to find out:

```
set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn_cv <- train(validity ~ ., method = "knn",
                      data = train_set,
                      tuneGrid = data.frame(k = seq(3, 51, 2)),
                      trControl = control)
ggplot(train_knn_cv, highlight = TRUE)
```



```
train_knn_cv$bestTune
```

```
##   k
## 2 5
```

```
y_hat5 <- predict(train_knn_cv,test_set)
mean5 <- mean(y_hat5 == test_set$validity)
confusionMatrix(data = y_hat5, reference = test_set$validity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 1626   78
##          2  385 1910
##
##                Accuracy : 0.8842
##                  95% CI : (0.8739, 0.894)
##     No Information Rate : 0.5029
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7686
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8086
##             Specificity : 0.9608
##          Pos Pred Value : 0.9542
##          Neg Pred Value : 0.8322
##              Prevalence : 0.5029
##          Detection Rate : 0.4066
##    Detection Prevalence : 0.4261
##       Balanced Accuracy : 0.8847
##
##        'Positive' Class : 1
##
```

```
results <- bind_rows(results,
                     data_frame(Method="KNN + Cross Validation",
                                Accuracy = mean5 ))
results %>% knitr::kable()
```

Method	Accuracy
Naive Bayes	0.8607152
LDA	0.9527382
QDA	0.8792198
GLM	0.9634909
KNN + Cross Validation	0.8842211

With the optimisation, the accuracy is significantly improved.

2.4.6 Random Forest

Finally, we investigate Random Forest. Random Forest expands on classification tree - it grows an ensenble of classification trees and derives the mean prediction from the classification results produced by different trees. It addresses decision tree's flaw of overfitting to the training set.

```r
set.seed(1, sample.kind = "Rounding")
train_rf<- train(validity ~ ., method ="rf", data = train_set, ntree=50, tuneGrid = data.frame(mtry=seq
train_rf$bestTune
```

```
##   mtry
## 7    7
```

```r
y_hat7 <- predict(train_rf,test_set)
mean7 <- mean(y_hat7 == test_set$validity)
varImp(train_rf) #let's inspect the most important
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 167)
##
##            Overall
## say         100.00
## video        90.16
## via          72.46
## image        47.98
## just         32.51
## like         32.22
## minister     29.10
## washington   27.96
## have         26.49
## don          25.86
## get          17.15
## wednesday    16.72
## america      14.95
## tuesday      14.56
## doesn        14.27
## watch        14.08
## wire         13.12
## thursday     13.10
## friday       12.90
## government   12.13
```

```r
confusionMatrix(data = y_hat7, reference = test_set$validity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 1952   78
##          2   59 1910
##
##                Accuracy : 0.9657
##                  95% CI : (0.9596, 0.9712)
##     No Information Rate : 0.5029
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9315
##
##  Mcnemar's Test P-Value : 0.1241
##
##             Sensitivity : 0.9707
##             Specificity : 0.9608
##          Pos Pred Value : 0.9616
```

```
##          Neg Pred Value : 0.9700
##              Prevalence : 0.5029
##          Detection Rate : 0.4881
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 0.9657
##
##        'Positive' Class : 1
##
```

```
results <- bind_rows(results,
                     data_frame(Method="Random Forest",
                                          Accuracy = mean7 ))
results %>% knitr::kable()
```

Method	Accuracy
Naive Bayes	0.8607152
LDA	0.9527382
QDA	0.8792198
GLM	0.9634909
KNN + Cross Validation	0.8842211
Random Forest	0.9657414

Random Forest is the best performig model out of all.

3. Results

The accuracy using the different model is as follows:

Method	Accuracy
Random Forest	0.9657414
GLM	0.9634909
LDA	0.9527382
KNN + Cross Validation	0.8842211
QDA	0.8792198
Naive Bayes	0.8607152

6 different methods have been tested, among which Random Forest out performs, arriving at an accuracy of 0.966.

4. Discussion and Conclusion

In this project, we first inspected the dataset and removed the subject column that would have caused biased results, then we performed textmining by harnessing the "tm" and "textstem" packages to remove uninformative words and return the meaningful, single base terms of words that are both stemmed and lemmatised.

Next, we inspect the frequencies and ratio of frequencies of those terms in the real and fake news sets, removing "Reuters" as it causes the dataset to be too biased as well as terms with a frequency too low or a ratio between 0.35-0.65 - author acknowledges the thresholds set for the frequency and ratio are arbitrary; this only presents a workaround for the limited computational power.

After the preprocessing, we performed different models - namely Naive Bayes, LDA, LQA, GLM, KNN and Random Forest on the datasets and compare the performance of each model by comparing the overall accuracy. Random Forest outperforms all models, arriving at an accuracy of 0.966.

The key lessons learned from this project are as follows:

- Data exploration and preprocessing is a crucial step for us to understand the nature, strengths and limitations of the dataset.
- It is important to remove elements that caused the dataset to be biased. If we did not remove the subject column as well as the term "Reuters", the models would be overfitting to this dataset. The aim of this project is to utilise text mining to derive terms that can be used as predictors for whether the article is real or fake.
- For this project, the key challenge does not lie in fitting the models, but at processing the data and performing appropriate text mining to allow the models to work on the substantial textual data and only utilise the important elements hidden between.

5. Bibliography

- Zarocostas, J. (2020). How to fight an infodemic. The Lancet, 395(10225), 676.
- Bisaillon, C (2020) Fake and real news dataset. Retrieved from https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
- Manning, C. and Schutze, H. (1999). Foundations of Statistical Natural Language Processing (1999) Retrieved from https://nlp.stanford.edu/fsnlp/
- Feinerer, I, Hornik, K. (2019) "tm" R package. Retrieved from https://cran.r-project.org/web/packages/tm/tm.pdf
- Rinker, T. (2018) "textstem" R package. Retrieved from https://cran.r-project.org/web/packages/textstem/index.html
- Bunkley, N (March 3, 2008) Joseph Juran, 103, Pioneer in Quality Control, Dies. The New York Times.
- McCallum, A (2019) Graphical Models, Lecture2: Bayesian Network Represention. Retrieved from https://people.cs.umass.edu/~mccallum/courses/gm2011/02-bn-rep.pdf
- Sckit Learn (2020) Linear and Quadratic Discriminant Analysis. Retrieved from https://scikit-learn.org/stable/modules/lda_qda.html
- Rokach, L; Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc. ISBN 978-9812771711.