Detecting Fake News Headlines Through Natural Language Processing

Yeonjoon Choi

09/04/2020

Introduction

Requirement

To run this Rmarkdown file, you will need to have the tm, naivebayes, wordcloud,ggplot2, tree, NLP caret,SnowballC,RColorBrewer, skmeans and e1071 package installed to your computer. Since I have specified every library as being required, you should be able to just run the file first.

Please note that the coed will not compile right away, so please give enough time for the codes to run.

Dataset and Background Information

The Onion is a satirical news site. This means that they publish fake news stories that are supposed to be funny. Some examples of headlines from The Onion are

-CDC Releases Instructions For All Americans To Make Their Own Hospitals

-Olympic Torchbearer Has Been Jogging In Place On Street Corner For Past 2 Weeks

-Exhausting Every Other Way To Pass Time, Couple Begins Ranking Their Lamps

www.reddit.com/r/nottheonion/ is an Internet Forum where people post real-life news articles that are so ridiculous that they are indistinguishable from articles from The Onion. Some examples of headlines posted in this forum are

-Driver caught doing 130mph claimed he was trying to avoid catching coronavirus

-Ben Affleck finally achieves lifelong dream of not having to play Batman anymore

-White House press secretary attacks media for accurately reporting inauguration crowds

The dataset contains headlines from The Onion, labeled 1, and headlines from r/NotTheOnion, labeled 0. The task is to correctly identify if a given headline is from The Onion, or real news.

The dataset was provided by Luke Feilberg in Github.

Technical Detail about the Dataset

Rather than using the dataset provided in the Github, I chose to generate my own dataset, following steps outlined by Luke Feilberg. This was because the dataset provided in the Github had some errors. However, the newly created dataset contains the same amount of headlines.

Also, the dataset is already cleaned to some extent. That is, the headlines are already all lower cases, and punctuations have already been removed.

Literature Review

Luke Feilberg provides a baseline model, which is created using Tensorflow. The model is a neural network, with LSTM cells. Feilberg achieves 85 % validation accuracy using this model.

My Goal

I intend to use models that do not fall under the umbrella of deep learning, which is different from what Feilberg did. Also, Feilberg did not provide the reader with other measures of their model, such as recall and precision. I will provide these evaluation metrics as to deeply analyze my own models.

Theoretical Background

Information in this section is from Natural Language Processing In Action by Hobson Lane.

In order to classify headlines through statistical models, we must first transform each headline into some kind of number. To do this, we employ the Bag-of-words model. This simply means that we disregard grammar and word order and view the dataset as a bag of words. For example, if we had the dataset,

I love life

I love pineapple pizza

Then our bag of words word would be {I, love, life, pineapple, pizza}

The corpus denotes all the unique words that appear in our headlines. So for example, {I, love, life, pineapple, pizza} was the corpus of the two sentences.

Our feature will be all unique words in the corpus of headlines. Each headline will correspond to a vector, where each element of vector is 0 or 1, with 0 meaning that the headline doesn't contain the specified word, and 1 meaning that the headline contains the specified word.

Let's do an example. Say our dataset was compromised of

Headline 1: man eats a lot of food

Headline 2: woman says she likes food

Then our corpus is {man, eats, a, lot, of, food, woman, says, she, likes}

Then our matrix will look like

Example 1:

So this means headline 1 corresponds to the vector (1,1,1,1,1,1, 0,0,0,0) and headline 2 corresponds to the vector (0,0,0,0,0,1,1,1,1,1)

One possible approach to model this data is to use Multivariate Bernoulli Naive Bayes.

In this model, we calculate $P(Onion|\vec{x}) \propto P(\vec{x}|Onion)P(Onion)$ and $P(NotOnion|\vec{x}) \propto P(\vec{x}|NotOnion)P(NotOnion)$, where $\vec{x}$ is the vector as above, and choose the class with the biggest posterior probability.

$P(Onion)$ and $P(NotOnion)$ can simply be estimated by the proportion of headlines that are Onion/Not Onion among all the headlines.

And from the independence assumption we have, with m being the size of the corpus and $b \in 0,1$, where $b = 1$ if x_i associated with 1 and 0 otherwise.

$$P(\vec{x}|Onion) = \prod_{i=1}^{m} P\,(x_i|Onion)^b(1 - P(x_i|Onion))^{1-b}$$

where $\hat{P}(x_i|Onion)$ is the maximum-likelihood estimate that a particular word x_i occurs in the class Onion, with

$$\hat{P}(x_i|Onion) = \frac{df_{xi,y} + 1}{df_y + 2}$$

where

$df_{xi,y}$ is the number of headlines in the training dataset that contains the feature x_i that belong to class Onion.

df_y is the number of headlines in the training dataset that belong to class Onion.

And of course the same can be done for the class Not Onion.

The above information was taken from the article Naive Bayes and Text Classification I, by Sebastian Raschka.

Analysis

Exploratory Data Analysis

```
#If you have the dataset downloaded in your workspace,
#please load the dataset as follows
```

```r
data = read.csv("OnionOrNotclean_V2.csv")

#If not, you can use below code after uncommenting it

#data =
read.csv("https://raw.githubusercontent.com/philiandsophia/Stat3850_Final_Pro
jecct/master/OnionOrNotClean_V2.csv")

#First column is useless
data = data[,-1]

#We will only use the first 5000 samples
#This is because using the whole dataset can take up to
#10 minutes
data = data[1:5000,]


#How many Onion and Non-Onion articles do we have?
length(which(data$label == 0))
```

```
## [1] 3150
```

```r
length(which(data$label == 1))
```

```
## [1] 1850
```

```r
#so we have 3150 non-Onion articles
#and 1850 Onion articles


#Let's create a word cloud of headlines that are Onion and headlines that are
not Onion
require(RColorBrewer)
```

```
## Loading required package: RColorBrewer
```

```r
require(wordcloud)
```

```
## Loading required package: wordcloud
```

```
## Warning: package 'wordcloud' was built under R version 3.6.3
```

```r
#for Onion headlines

wordcloud(data$text[data$label == 1], min.freq = 20,random.order = FALSE)
```

```
## Loading required namespace: tm
```

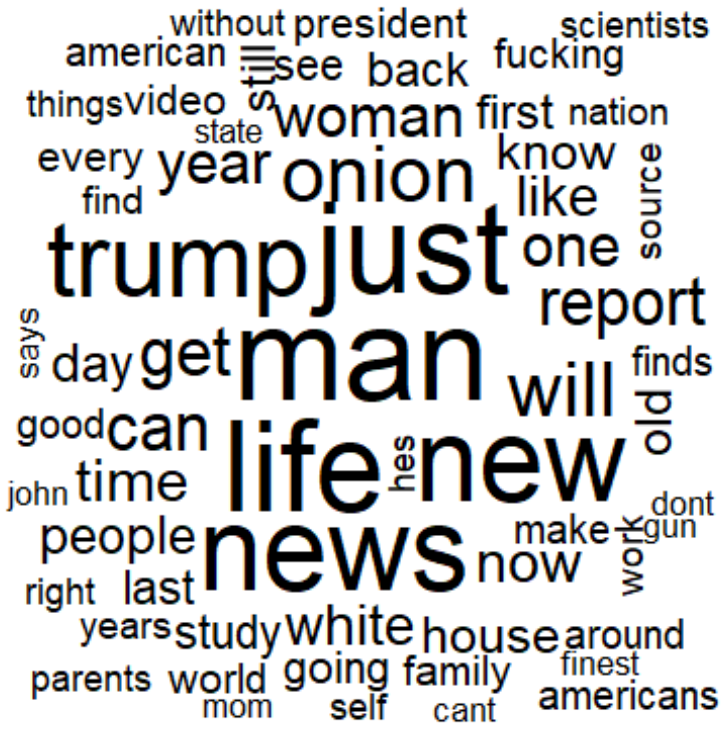
```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents

## Warning in wordcloud(data$text[data$label == 1], min.freq = 20,
random.order =
## FALSE): campaign could not be fit on page. It will not be plotted.

## Warning in wordcloud(data$text[data$label == 1], min.freq = 20,
random.order =
## FALSE): americas could not be fit on page. It will not be plotted.

## Warning in wordcloud(data$text[data$label == 1], min.freq = 20,
random.order =
## FALSE): getting could not be fit on page. It will not be plotted.
```

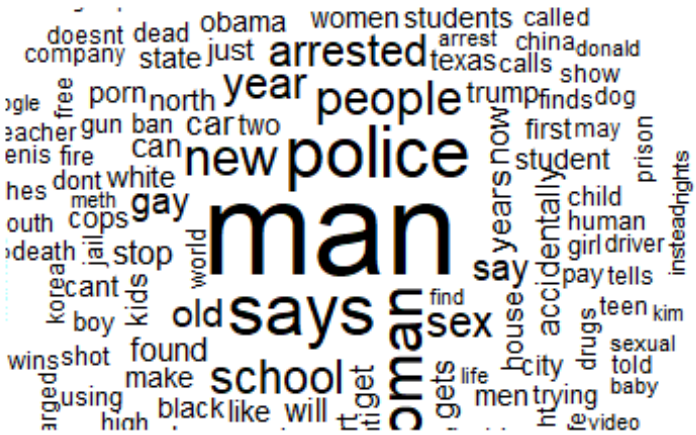



```
#for non-Onion headlines

wordcloud(data$text[data$label == 0], min.freq = 20,random.order = FALSE)

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
transformation
## drops documents

## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```



Multivariate Bernoulli Naive Bayes Model

The R package tm and e1071 will be used.

The analysis procedure will be as follows.

-Clean headlines by removing numbers, stop words and stemming, using the tm package

-Using the tm package, transform the dataset into a document term matrix, which is a matrix constructed as in Example 1 in the theoretical background section.

-Only keep words that appear at least 5 times. This will reduce dimensionality, and the information loss is minimum since words that appear less than 5 times do not carry much information.

-Use e1071 library to construct Multivariate Bernoulli Naive Bayes Model

Let's first clarify what I mean by stop word and stemming.

Stop words: Words like The, a, in, and of do not add much to the meaning of the headlines. Hence, we remove them from each headline.

Stemming: Words such as learning, learned, learn have almost identical meanings. Hence, we reduce the three words to learn, the stem word.

```
#If you have the dataset downloaded in your workspace,
#please load the dataset as follows
data = read.csv("OnionOrNotclean_V2.csv")
```

```r
#If not, you can use below code after uncommenting it

#data =
read.csv("https://raw.githubusercontent.com/philiandsophia/Stat3850_Final_Project/master/OnionOrNotClean_V2.csv")

#First column is useless
data = data[,-1]

#We will only use the first 5000 samples
#This is because using the whole dataset can take up to
#10 minutes
data = data[1:5000,]

#Let's take a peek at the dataset
head(data)
```

```
##
text
## 1
video at last man shows his pet tarantula a maraschino cherry
## 2
missing plane lost malaysia says
## 3 do you need us to hold your hand through this entire list of incredible
murals do we always have to compile a whole list of photos for you just so
that youll look at them grow up
## 4
man dressed as elsa from frozen pushes boston police wagon out of snow
## 5
56 year old is slammed for controversial form of baby yoga
## 6
it is with a heavy heart that i announce i am having my parents pick me up
early from this sleepover
##   label
## 1     1
## 2     0
## 3     1
## 4     0
## 5     0
## 6     1
```

```r
#load the tm library
require(NLP)
```

```
## Loading required package: NLP
```

```r
require(tm)
```

```
## Loading required package: tm
```

```
## Warning: package 'tm' was built under R version 3.6.3

#define our corpus, set of all unique words
corpus = VCorpus(VectorSource(data$text))

#create the document term matrix, as explained above.
#We remove the stop words, numbers and we perform stemming.
require(SnowballC)

## Loading required package: SnowballC

## Warning: package 'SnowballC' was built under R version 3.6.3

dtm = DocumentTermMatrix(corpus, control = list(stopwords = TRUE, stemming =
TRUE, removeNumbers = TRUE))

#split the dataset into training set and test set
set.seed(1243)
idx = sample(1:5000, size = 500)

train = dtm[-idx,]
train_label = data$label[-idx]

test = dtm[idx,]
test_label = data$label[idx]

#We will only keep words that appear at least 5 times.
freq = findFreqTerms(train, 5)

#The number of features will be
length(freq)

## [1] 1854

#Manipulate the train and test matrix
#so only words that appear 5 times are kept
train = as.matrix(train)
test = as.matrix(test)
train = train[, freq]
test = test[,freq]

#We want to reat 0 and 1 as categorical variables,
#not numerical variables
#Hence, we change them into strings
train[train == 0] = "NotIn"
train[train == 1] = "In"

test[test == 0] = "NotIn"
test[test == 1] = "In"
```

```r
#load e1071 library, construct naive Bayes model
require(e1071)
```

```
## Loading required package: e1071
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```r
classifier = naiveBayes(train, as.factor(train_label))

#prediction on test data
pred = predict(classifier, test)

#the confusion matrix is given by
conft = table(predicted = pred, actual = as.factor(test_label))
conft
```

```
##           actual
## predicted   0   1
##         0 275  56
##         1  47 122
```

```r
#Evaluation metrics can be calculated as
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```r
require(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```r
sensitivity(conft)
```

```
## [1] 0.8540373
```

```r
specificity(conft)
```

```
## [1] 0.6853933
```

```r
precision(conft)
```

```
## [1] 0.8308157
```

Multinomial Naive Bayes Model

Now, we will use the Naive Bayes Model, where we assume each vector is from a multinomial distribution. Instead of using the document term matrix like above, we will use Term Frequency - Inverse Document Frequency matrix(tf–idf). This matrix characterizes how important a word is to a headline in the whole corpus. Our analysis will go as below.

-Clean headlines by removing numbers, stop words and stemming, using the tm package

-Using the tm package, transform the dataset into tf–idf matrix.

-Only keep words that appear at least 5 times. This will reduce dimensionality, and the information loss is minimum since words that appear less than 5 times do not carry much information.

-Use naivebayes library to construct Multinomial Naive Bayes Model

```
#If you have the dataset downloaded in your workspace,
#please load the dataset as follows
data = read.csv("OnionOrNotclean_V2.csv")

#If not, you can use below code after uncommenting it

#data =
read.csv("https://raw.githubusercontent.com/philiandsophia/Stat3850_Final_Pro
jecct/master/OnionOrNotClean_V2.csv")

#First column is useless
data = data[,-1]

#We will only use the first 5000 samples
#This is because using the whole dataset can take up to
#10 minutes
data = data[1:5000,]

#load the tm library
require(NLP)
require(tm)

#define our corpus, set of all unique words
corpus = VCorpus(VectorSource(data$text))

require(SnowballC)
dtm = DocumentTermMatrix(corpus, control = list(stopwords = TRUE, stemming =
TRUE, removeNumbers = TRUE, termFreq = TRUE))

#create tf-idf matrix
dtm = weightTfIdf(dtm)
```

```
## Warning in weightTfIdf(dtm): empty document(s): 3293
```

```
set.seed(1243)
idx = sample(1:5000, size = 500)

#create training set and testing set
train = dtm[-idx,]
train_label = data$label[-idx]

test = dtm[idx,]
test_label = data$label[idx]


#We will only keep words that appear at least 5 times.
freq = findFreqTerms(train, 5)

#The number of features will be
length(freq)
```

```
## [1] 1992
```

```
#Manipulate the train and test matrix
#so only words that appear 5 times are kept
train = as.matrix(train)
test = as.matrix(test)
train = train[, freq]
test = test[,freq]


require(naivebayes)
```

```
## Loading required package: naivebayes
```

```
## Warning: package 'naivebayes' was built under R version 3.6.3
```

```
## naivebayes 0.9.7 loaded
```

```
#create multinomial naive bayes model
classifier = multinomial_naive_bayes(train, as.factor(train_label), laplace =
1)
#predict on test data
pred = predict(classifier, test)
#the confusion matrix is given by
conft = table(predicted = pred, actual = as.factor(test_label))
conft
```

```
##           actual
## predicted   0   1
##         0 272  63
##         1  50 115
```

```r
#Evaluation metrics can be calculated as
require(ggplot2)
require(caret)

sensitivity(conft)

## [1] 0.8447205

specificity(conft)

## [1] 0.6460674

precision(conft)

## [1] 0.8119403
```

Result and Comparsions

Other failed methods

Clustering Methods

Since ti-idf matrix measures how similar each headline is, I tried using clustering method to cluster similar headlines together. Since Natural Language Processing In Action by Lane recommended that cosine similarity measure should be used in comparing vectors in ti-idf matrices, I chose to use the spherical k-means method, instead of k-means method which uses Euclidean distance. This attempt was not successful.

```r
#Format the data into tf-idf matrix as before
data = read.csv("OnionOrNotclean_V2.csv")

#data =
read.csv("https://raw.githubusercontent.com/philiandsophia/Stat3850_Final_Project/master/OnionOrNotClean_V2.csv")


data = data[,-1]
data = data[1:5000,]

require(NLP)
require(tm)

corpus = VCorpus(VectorSource(data$text))
require(SnowballC)
dtm = DocumentTermMatrix(corpus, control = list(stopwords = TRUE, stemming =
TRUE, removeNumbers = TRUE, termFreq = TRUE))

dtm = weightTfIdf(dtm)
```

```
## Warning in weightTfIdf(dtm): empty document(s): 3293
```

```
set.seed(1243)
idx = sample(1:5000, size = 500)

test = dtm[idx,]
test_label = data$label[idx]
```

```
require(skmeans)
```

```
## Loading required package: skmeans
```

```
## Warning: package 'skmeans' was built under R version 3.6.3
```

```
set.seed(123444)
#cluster into two groups, hopefully
#we group the Onion headlines with Onion headlines
#and non-Onion headlines with non-Onion headlines

#we use Spherical k-means, with 2 clusters.
km_out = skmeans(test, 2)


#But since no matter how we label the cluster
#we do not get good result
#so it seems we didn't cluster Onion headlines with
#Onion headlines
#and non-Onion headlines with non-Onion headlines
cluster = km_out$cluster
cluster[cluster == 1] = 1
cluster[cluster == 2] = 0

table(predicted = cluster, actual = test_label)
```

```
##          actual
## predicted   0   1
##         0 153  86
##         1 169  92
```

```
cluster = km_out$cluster
cluster[cluster == 1] = 0
cluster[cluster == 2] = 1

table(predicted = cluster, actual = test_label)
```

```
##          actual
## predicted   0   1
##         0 169  92
##         1 153  86
```

I have also tried using tree model on the document term matrix, but this did not produce good result

```
#Format the data as before,
#but this time we need to make the data
#into a dataframe
data = read.csv("OnionOrNotclean_V2.csv")

#data =
read.csv("https://raw.githubusercontent.com/philiandsophia/Stat3850_Final_Project/master/OnionOrNotClean_V2.csv")


data = data[,-1]
data = data[1:5000,]

require(NLP)
require(tm)

corpus = VCorpus(VectorSource(data$text))
require(SnowballC)
dtm = DocumentTermMatrix(corpus, control = list(stopwords = TRUE, stemming = TRUE, removeNumbers = TRUE))

set.seed(1243)
idx = sample(1:5000, size = 500)

train = dtm[-idx,]
train_label = data$label[-idx]

test = dtm[idx,]
test_label = data$label[idx]


freq = findFreqTerms(train, 5)

train = as.matrix(train)
test = as.matrix(test)
train = train[, freq]
test = test[,freq]

train[train == 0] = "NotIn"
train[train == 1] = "In"

test[test == 0] = "NotIn"
test[test == 1] = "In"

#make the document term matrix into a dataframe
#that contains the labels
```

```r
train = cbind(train, train_label)
train = as.data.frame(train)

test = cbind(test, test_label)
test = as.data.frame(test)

#We have to rename the columns
#to make the code work
columns = c()

for (i in 1:1854){
  columns = c(columns, sprintf("x%d",i))
}

names(train) = c(columns, "y")
names(test) = c(columns, "y")



require(tree)
```

```
## Loading required package: tree
```

```
## Warning: package 'tree' was built under R version 3.6.3
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```r
classifier = tree(y~., data = train)
#We do not produce good result
pred = predict(classifier, test, type = "class")
table(predicted = pred, actual = as.factor(test_label))
```

```
##          actual
## predicted   0   1
##         0   0   0
##         1 322 178
```

Result and Comparison between two Naive Bayes Model

Overall, only the two Naive Bayes model were successful. From the confusion matrix produced, we can compare the results of the two Naive Bayes models.

The results on the testing data can be summarized as(Note that Onion headlines are treated as negative in this case)

	Multivariate Bernoulli Naive Bayes	Multinomial Naive Bayes
accuracy	0.7940	0.7740
sensitivity	0.8540	0.8447
specificity	0.6854	0.6461
precision	0.8308	0.8119

Looking at this table it is clear that the multivariate Bernoulli Naive Bayes model is better than the multinomial Naive Bayes model, as the first outperforms the latter in all evaluation metrics.

Conclusion

Out of the two good models,the multivariate Bernoulli Naive Bayes model was better. With sensitivity of 0.8540, this means that given that the headline was not from The Onion, the model predicted correctly about 85% of the time. With specificity of 0.6854, this means given that the headline was from The Onion, the model predicted correctly about 68% of the time. Precision of 0.8308 means that given the model predicted the headline as not an Onion article, this prediction will be right about 83% of the time.

It does seem surprising that the model has a harder time predicting Onion headlines. Since Onion headlines are supposed to be funny, humans can easily identify Onion headlines. However, we must consider that our samples contained fewer Onion headlines compared to not Onion headlines. This means, given more Onion headlines, we could train our models to be better at predicting Onion headlines.

The accuracy is low compared to the baseline model provided by Feilberg, but this is to be expected as we used less samples.

What I have learned

The major takeaway from this project has been learning to work with the tm package. I was able to learn about a major part in natural language processing, which is cleaning data through methods such as removing stop words and stemming.

I also learned how to transform dataset of natural languages into numerical values, and how to apply statistical models to them. I learned about the bag-of-words model and the theory behind how to apply the Naive Bayes model to classify based on our dataset.

I also learned how to create word clouds through the wordcloud library. This will be useful skill to have in terms of data visualization.

Despite it being a failed attempt, I learned about Spherical k-mean clustering, and how it may be applied through the skmeans library.

I realize that I have only scratched the surface of natural language processing, and I intend to do a deeper dive in future projects.

References

Feilberg, Luke. “Lukefeilberg/Onion.” GitHub, 25 Feb. 2020, github.com/lukefeilberg/onion.

Feinerer , Ingo. “Package ‘Tm’.” Tm: Text Mining Package, cran.r-project.org/web/packages/tm/tm.pdf.

Lane, Hobson, et al. Natural Language Processing in Action Understanding, Analyzing, and Generating Text with Python. Manning Publications Company, 2019.

Raschka, Sebastian. “Naive Bayes and Text Classification I - Introduction and Theory.” ArXiv.org, 14 Feb. 2017, arxiv.org/abs/1410.5329.

Links to all online references

-https://github.com/lukefeilberg/onion

-https://arxiv.org/abs/1410.5329

-https://cran.r-project.org/web/packages/tm/tm.pdf