Data Science Capstone Milestone Report

Jacky Wong

Tuesday, March 24, 2015

Executive Summary

This project involves making prediction on the possible next word that may appear given a single word or a phrase that the user enters. Such a prediction application is used in Mobile Devices as user types and even in search boxes of search engines which can predict and suggest words as one type into the search boxes. This milestone report look at the approach in creating the application, including downloading the dataset, performing data exploration, preprocessing the data and preparing the data for prediction models.

Data Collection

Data were downloaded from the course website, which comprises of Blogs, Twitters and News text. These would serve as corpora for developing the prediction model.

Basic Data Exploration

File sizes of the 3 source files in MB:

```
# file size
file.info("data/en_US/en_US.blogs.txt")$size   / 1024^2
```

```
## [1] 200.4
```

```
file.info("data/en_US/en_US.news.txt")$size    / 1024^2
```

```
## [1] 196.3
```

```
file.info("data/en_US/en_US.twitter.txt")$size / 1024^2
```

```
## [1] 159.4
```

Number of lines of each data:

```
# number of lines
length(blogs)
```

```
## [1] 899288
```

```
length(news)
```

```
## [1] 1010242
```

```
length(twitter)
```

```
## [1] 2360148
```

Number of characters per line:

```
# number of characters per line
summary( nchar(blogs)   )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1      47     156     230     329   40800
```

```
summary( nchar(news)    )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     110     185     201     268   11400
```

```
summary( nchar(twitter) )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     2.0    37.0    64.0    68.8   100.0   213.0
```

```
## Warning: package 'stringi' was built under R version 3.1.3
```

Analysis of empty lines and characters that are not white space:

```
# Blogs
stats_blogs
```

```
##       Lines LinesNEmpty       Chars CharsNWhite 
##      899288      899288   206824382   170389539
```

```
# News
stats_news
```

```
##       Lines LinesNEmpty       Chars CharsNWhite 
##     1010242     1010242   203223154   169860866
```

```
# Twitter
stats_twitter
```

```
##       Lines LinesNEmpty       Chars CharsNWhite 
##     2360148     2360148   162384825   134370070
```

Summary of textual data characteristics:

```
# summaries
summary( words_blogs   )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       9      28      42      60    6730
```

```
summary( words_news    )
```

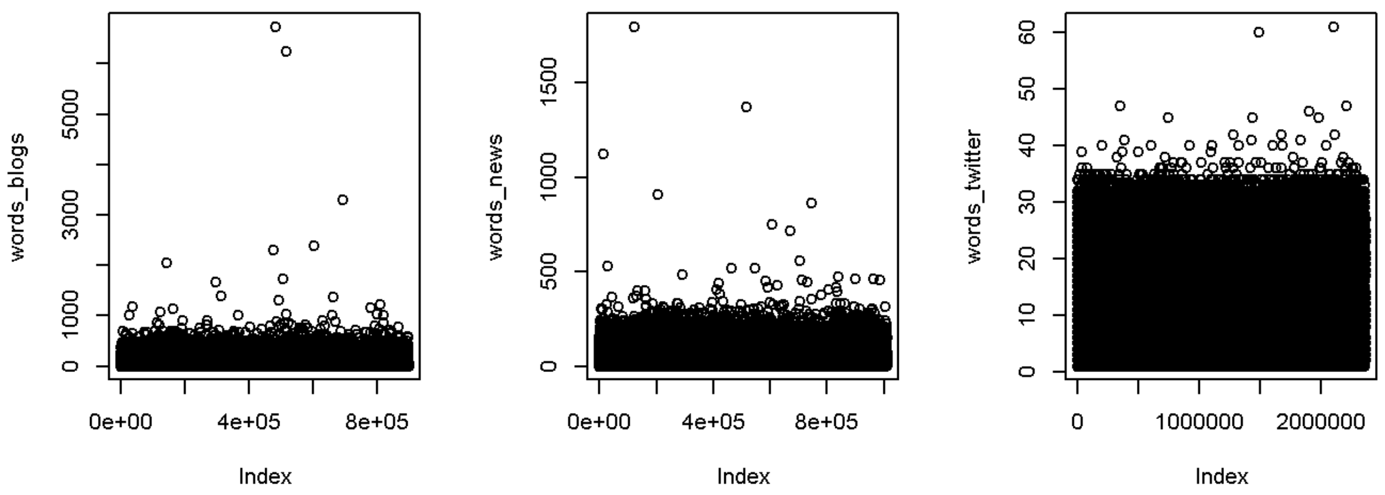
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    19.0    32.0    34.4    46.0  1800.0
```

```
summary( words_twitter )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     7.0    12.0    12.8    18.0    61.0
```

Plots of the textual data characteristics:

```
par(mfrow=c(1,3))
plot(words_blogs)
plot(words_news)
plot(words_twitter)
```



Summary of Data Exploration

Blogs and News are similar in terms of file size, number of lines, mean and median number of words per line. However, blogs is generally using more words than a typical news data source. Twitter on the other hand, displays very distinct characteristics, the mean and median number of words are a lot less than words and blogs. I think combining the data sources make sense because doing so would cater to different language style, for example, the formal reporting style of news, the less formal but wordy nature of blogs and lastly, short form nature of twitter.

Preprocessing Data

As the data source is large, I chose to sample 100,000 from each data (blogs,news,twitter). From this set, I combined the data sources.I performed the following processing on the combined data: 1. Change all to lower case. 2. Remove blank spaces. 3. Remove numbers. 4. Change & to " and " 5. Remove symbols / punctuations.

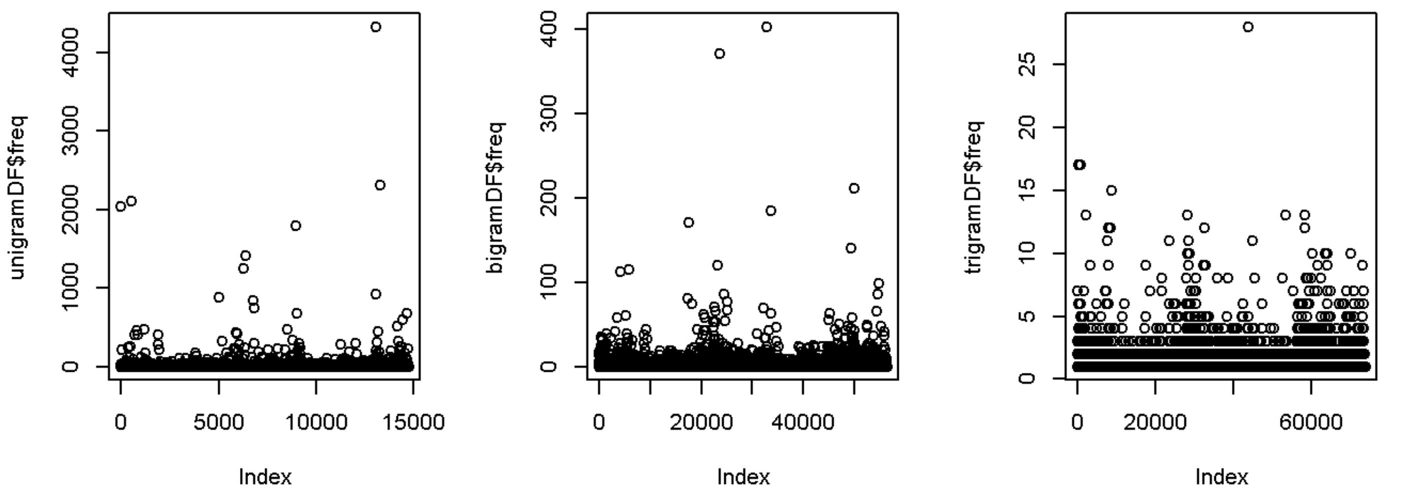
Tokenization

Using RWeka package, I construct 3 data frames from the processed data source: 1. Unigram Dataframe - dataframe with unique terms and their frequency. 2. Bigram Dataframe - dataframe with two terms and their frequency. 3. Trigram Dataframe - dataframe with three terms and their frequency.

The idea to create these N-Gram data frames and look at the frequency of occurance to produce a statistically meaningful prediction model.

Plots of the n-gram data frames frequency characteristics:

```
par(mfrow=c(1,3))
plot(unigramDF$freq)
plot(bigramDF$freq)
plot(trigramDF$freq)
```



From the plots, although most of the terms do not have high frequency, those bigram and trigram that does have high frequency would serve as good predictor.

Prediction Models

The approach to build the prediction models is such that given an input of two words, e.g. “have to”, the prediction model will consult the Trigram Dataframe, look at the highest frequency of a trigram with “have to” and produce the top few words based on the frequency of occurance. For the example “have to”, consulting trigram data frame will show the following:

Trigram	**Frequency**
“have to get”	5
“have to be”	4
“have to say”	3

The prediction model shall predict that the next word after “have to” would be “get”,“be” and “say” in that order.

Another possible approach is that given the N-gram data frame, I can train the machine to learn using machine learning model such as Naive Bayes, K-Nearest Neighbor or Decision Tree and then run the prediction using the trained model. Given the memory constraints imposed, some amount of trial and errors would have to be done to come up with an efficient and accurate prediction model.

Look forward to your feedback and comment.