Data exploration

TD

Executive summary

In this report the data of the twitter, blogs and news documents are explored. The total amount of data was seen to be large so sub-sampling is necessary. Three n-grams seems to be the largest number of usefull ngrams

Setting upt the environment and loading data

The environment is setup by loading the stringr, dplyr, tm, stringi and RWeka libraries.

```
library(stringr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tm)
```

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

```
library(stringi)
library(RWeka)
rm(list=ls())
```

Next data is loaded from the files

```
print(getwd())
```

```
## [1] "C:/Users/tdurieux/Documents/Repositories/coursera_repos/Coursera_hopkins_capstone/02_exploratory
```

```
# TWITTER
con <- file("Data/en_US.twitter.txt", "r")
twitter <- readLines(con, skipNul = TRUE)
close(con)

# BLOGS
con <- file("Data/en_US.blogs.txt", "r")
blogs <- readLines(con, skipNul = TRUE)
close(con)

# NEWS
con <- file("Data/en_US.news.txt", "r")
news <- readLines(con, skipNul = TRUE)
close(con)
```

Initial exploration

A first exploration shows that the twitter file contains over 2.3 million tweets, the blog almost 0.9 million blogs and the news file only 77 news articles. On the other hand the twitter file is also far shorter per item To make ngrams it seems better to use more of the news and blog files than the twitter file.

```
# Count lines per file
print(paste('The twitter file contains ', length(twitter), ' lines'))
```

```
## [1] "The twitter file contains  2360148  lines"
```

```
print(paste('The news file contains ', length(news), ' lines'))
```

```
## [1] "The news file contains  77259  lines"
```

```
print(paste('The blogs file contains ', length(blogs), ' lines'))
```

```
## [1] "The blogs file contains  899288  lines"
```

```
# Count words per file
print(paste(
    'The twitter file contains ', sum(stri_count_words(twitter)), ' words',
    ' with an average of ', mean(stri_count_words(twitter)), ' per tweet'
    ))
```

```
## [1] "The twitter file contains  30218166  words  with an average of  12.8035046954683  per tweet"
```

```
print(paste(
    'The news file contains ', sum(stri_count_words(news)), ' words',
    ' with an average of ', mean(stri_count_words(news)), ' per article'
    ))
```

```
## [1] "The news file contains  2693898  words  with an average of  34.8684036811245  per article"
```

```
print(paste(
    'The blog file contains ', sum(stri_count_words(blogs)), ' words',
    ' with an average of ', mean(stri_count_words(blogs)), ' per blog'
    ))
```

```
## [1] "The blog file contains  38154238  words  with an average of  42.4271623773474  per blog"
```

A corpus is made by combining samples of equal numbers of items

```
set.seed(42)
data_sample <- c(sample(twitter, 1000),
                 sample(blogs, 1000),
                 sample(news, 1000))
rm(twitter, blogs, news)
```

And a vcorpus is made

```
corpus <- VCorpus(VectorSource(data_sample))
```

In cleaning it is found that there are file names that can be removed, twitter handles and hashtags. Next non-regular characters are removed, non-letters, everything is put to lower, numbers are removed, whitespace stripped, stopwords are removed, etc.

```
replace_string <- function(x, pattern) gsub(pattern, " ", x)
replace_regex <- function(x, pattern) str_replace_all(x, pattern, " ")
delete_regex <- function(x, pattern) str_replace_all(x, pattern, "")
replace_contraction_error <- function(doc) {
```

```
    doc <- gsub("nÃ¢???Tt", "n't", doc)
    doc <- gsub("Ã¢???Tll", "'ll", doc)
    doc <- gsub("Ã¢???Tre", "'re", doc)
    doc <- gsub("Ã¢???Tve", "'ve", doc)
    doc <- gsub("Ã¢???Tm", "'m", doc)
    doc <- gsub("itÃ¢???Ts", "it's", doc) # a special case of 's
    return(doc)
}


clean_string <- function(x){
    x <- tolower(x)
    x <- replace_contraction_error(x)
    x <- removeWords(x, stopwords("en"))
    x <- replace_string(x, "(f|ht)tp(s?)://(.*)[.][a-z]+")  # remove files
    x <- replace_string(x, "@[^\\s]+")  # remove twitter handles
    x <- replace_regex(x, "#[\\w|\\d]*")  # remove hashtags
    x <- delete_regex(x, "[^\\w|\\s]*")  # remove everything but letters and whitespace
    x <- replace_regex(x, "[^ -~]")  # remove none utf-8 characters
    x <- removeNumbers(x)
    x <- stripWhitespace(x)
    x <- delete_regex(x, "^\\s*")  # remove leading whitespaces
    x <- delete_regex(x, "\\s*$")  # remove trailing witespaces
    x
}
```

The cleaning function is used to clean the corpus

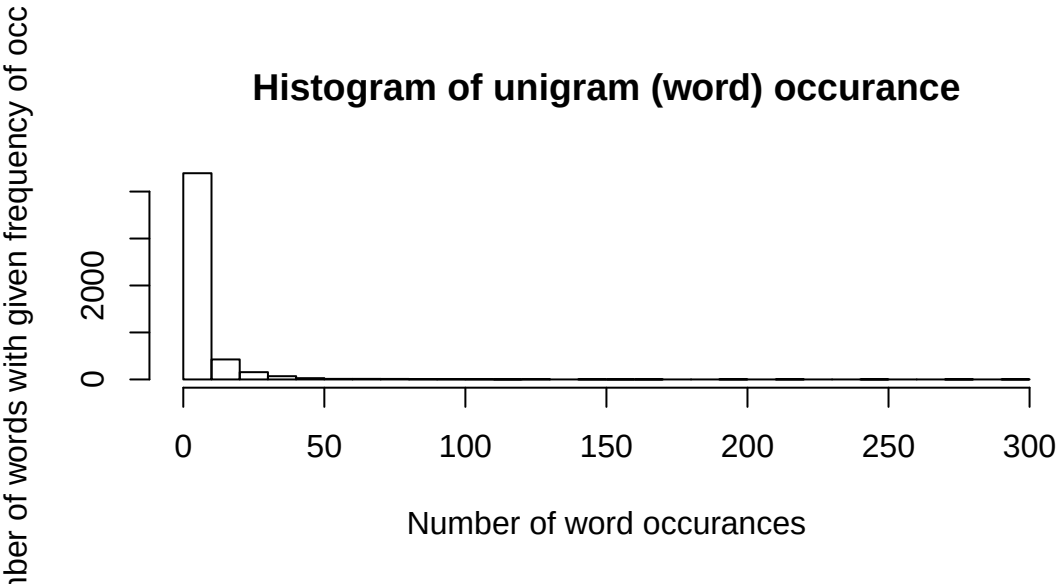
```
clean_corpus <- tm_map(corpus, clean_string)
clean_corpus <- tm_map(clean_corpus, PlainTextDocument)
# rm(corpus)
```

It can be seen that there are unigrams (words) that have up to 299 different occurences (remember, after removing stopwords). 'said' and 'will' are the one with the most occurences.

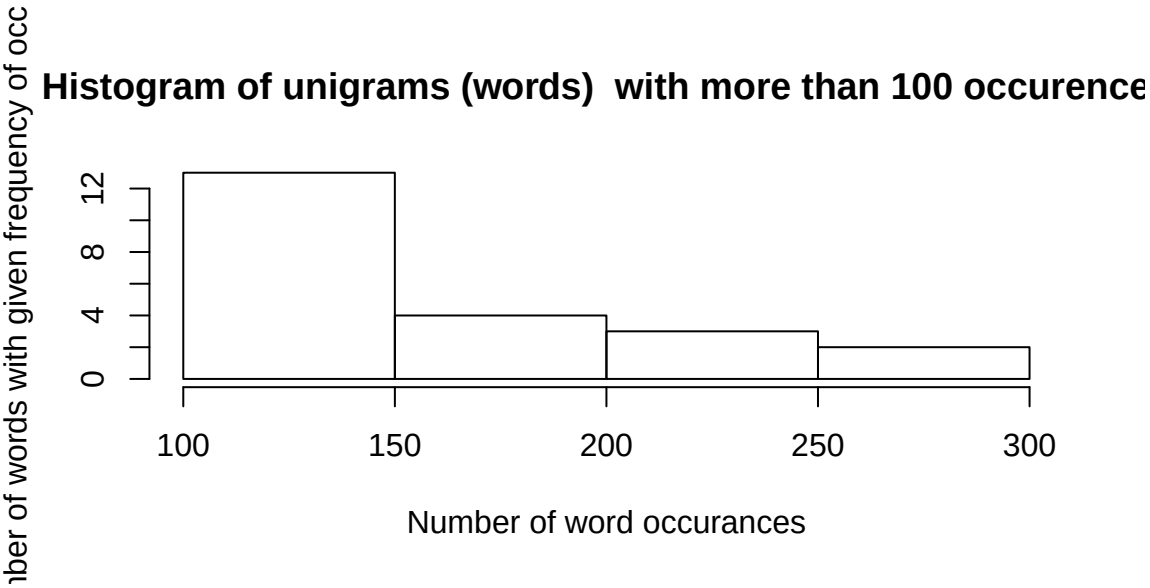
```
unigram_tokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))

dtm_unigram <- DocumentTermMatrix(clean_corpus, control = list(tokenize = unigram_tokenizer))
n_docs <- nDocs(dtm_unigram)
dtm_unigram <- removeSparseTerms(dtm_unigram, (n_docs - 1.1) / n_docs)

tbl_dtm_unigram <- as_tibble(as.matrix(dtm_unigram))
tbl_frequencies <- summarise_all(tbl_dtm_unigram, funs(sum))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## please use list() instead
##
##   # Before:
##   funs(name = f(.))
##
##   # After:
##   list(name = ~ f(.))
## This warning is displayed once per session.
```

```r
unigram_frequencies <- unlist(tbl_frequencies)

rm(dtm_unigram)
rm(tbl_dtm_unigram)
rm(tbl_frequencies)

hist(
    unigram_frequencies, breaks = 30,
    xlab='Number of word occurances',
    ylab='Number of words with given frequency of occurance',
    main='Histogram of unigram (word) occurance'
)
```




```r
hist(
    unigram_frequencies[which(unigram_frequencies > 100)],
    xlab='Number of word occurances',
    ylab='Number of words with given frequency of occurance',
    main='Histogram of unigrams (words)  with more than 100 occurences'
)
```




```
print('the unigrams with more than 200 occurences are:')
```

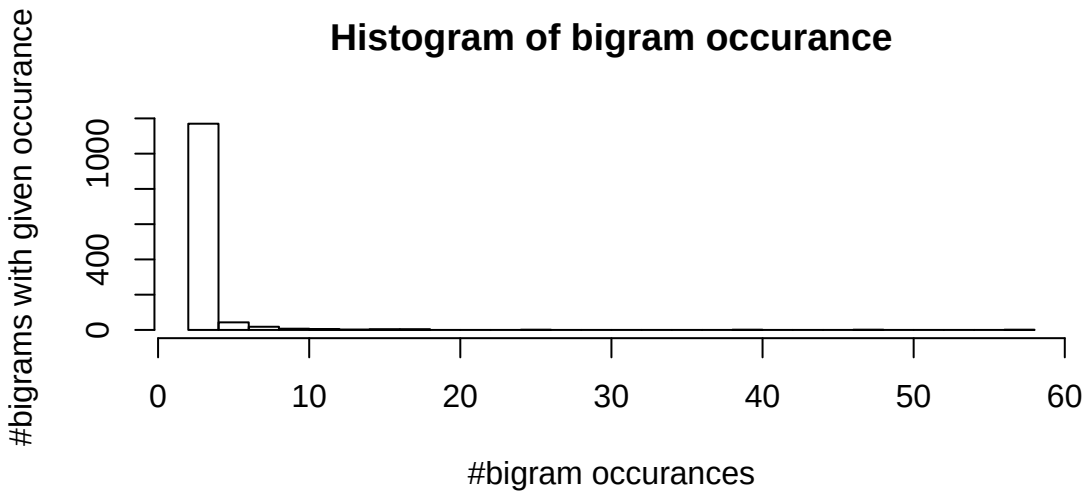
```
## [1] "the unigrams with more than 200 occurences are:"
```

```
print(unigram_frequencies[which(unigram_frequencies > 200)])
```

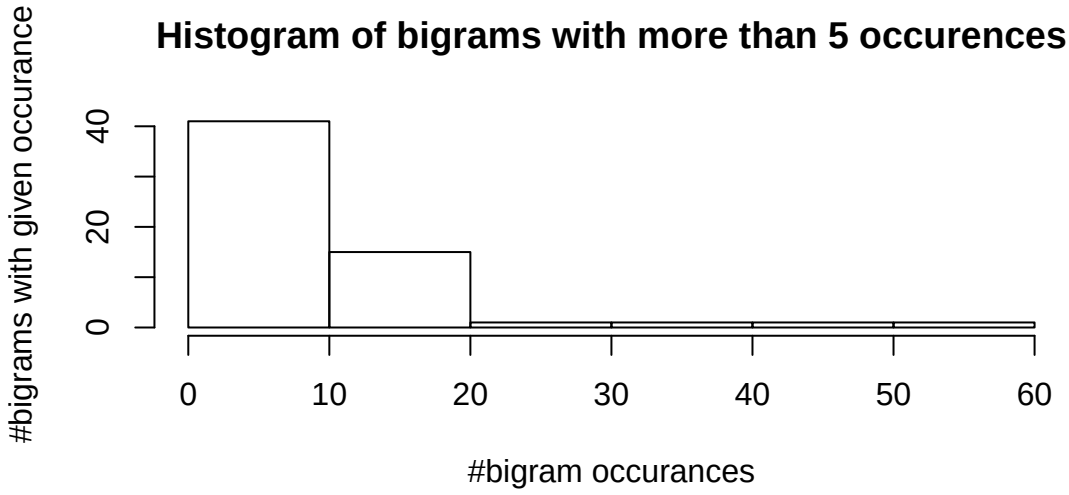
```
## just like  one said will
##  219  215  246  299  272
```

Most bigrams have viewer than 3 occurances (we left the 1 occurence out). There are 5 occurences of bi-grams with more than 15 occurences. All seems logical like 'new york' and 'right now'.

```
bigram_tokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
dtm_bigram <- DocumentTermMatrix(clean_corpus, control = list(tokenize = bigram_tokenizer))
n_docs = nDocs(dtm_bigram)
dtm_bigram <- removeSparseTerms(dtm_bigram, (n_docs - 1.1) / n_docs)

tbl_dtm_bigram <- as_tibble(as.matrix(dtm_bigram))
tbl_frequencies <- summarise_all(tbl_dtm_bigram, funs(sum))
bigram_frequencies <- unlist(tbl_frequencies)

rm(dtm_bigram)
rm(tbl_dtm_bigram)
rm(tbl_frequencies)

hist(
    bigram_frequencies, breaks = 30,
    xlab='#bigram occurances',
    ylab='#bigrams with given occurance',
    main='Histogram of bigram occurance'
)
```




```
hist(
    bigram_frequencies[which(bigram_frequencies > 5)],
    xlab='#bigram occurances',
    ylab='#bigrams with given occurance',
    main='Histogram of bigrams with more than 5 occurences'
)
```




```
print('the bigrams with more than 15 occurences are:')
```

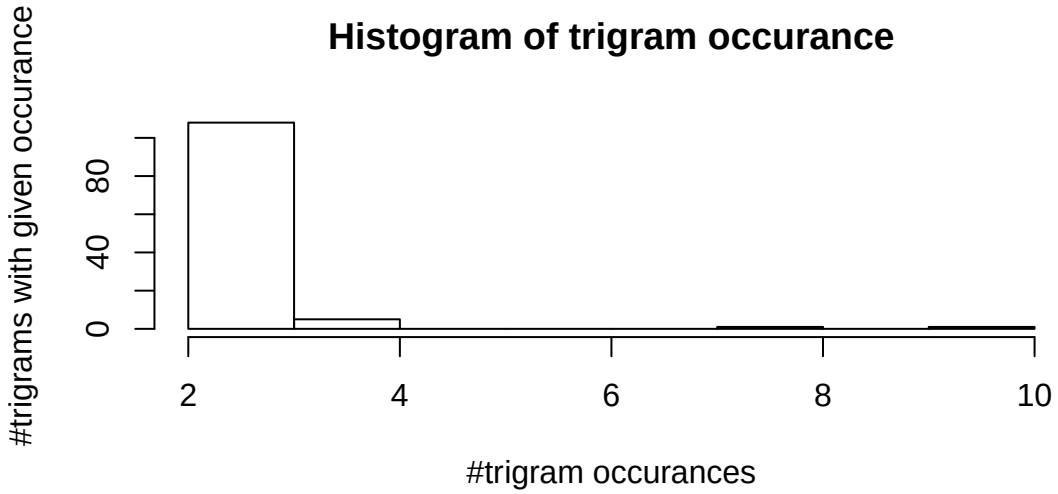
```
## [1] "the bigrams with more than 15 occurences are:"
```

```
print(bigram_frequencies[which(bigram_frequencies > 15)])
```

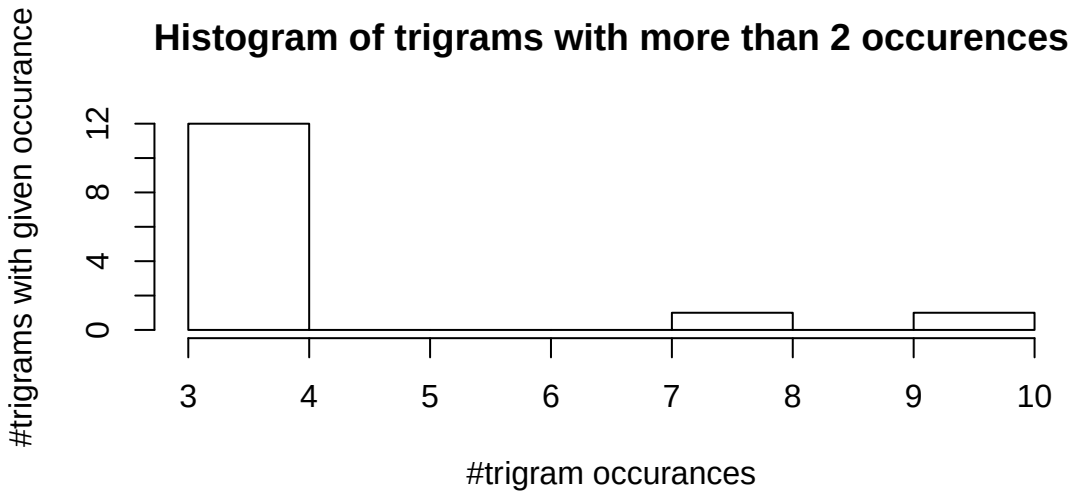
```
##   doesn t     don t       i m      it s last week last year make sure
##        17        39        57        47        18        17        16
##  new york right now
##        17        25
```

There are even 80 trigrams that occure twice and 6 that occure even more often.

```
trigram_tokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
dtm_trigram <- DocumentTermMatrix(clean_corpus, control = list(tokenize = trigram_tokenizer))

n_docs = nDocs(dtm_trigram)
dtm_trigram <- removeSparseTerms(dtm_trigram, (n_docs - 1.1) / n_docs)

tbl_dtm_trigram <- as_tibble(as.matrix(dtm_trigram))
tbl_frequencies <- summarise_all(tbl_dtm_trigram, funs(sum))
trigram_frequencies <- unlist(tbl_frequencies)

hist(
    trigram_frequencies,
    xlab='#trigram occurances',
    ylab='#trigrams with given occurance',
    main='Histogram of trigram occurance'
)
```




```
hist(
    trigram_frequencies[which(trigram_frequencies > 2)],
    xlab='#trigram occurances',
    ylab='#trigrams with given occurance',
    main='Histogram of trigrams with more than 2 occurences'
)
```




```
print('the trigrams with more than 2 occurences are:')
```

```
## [1] "the trigrams with more than 2 occurences are:"
```

```
print(trigram_frequencies[which(trigram_frequencies > 2)])
```

```
## amazon services llc          don t know          don t want
##                   4                   8                   3
##         g protein g      happy new year           i m going
##                   3                   4                  10
##           i m sure           it s just           know it s
##                   3                   3                   3
##       llc amazon eu        people don t services llc amazon
##                   4                   4                   4
##          think it s     three years ago
##                   3                   3
```

In this dataset there are only 2 quadgram with more than 2 occurances, both seem to be the Amazon company name. Quadgrams therefore seem to be too much.

```
quadgram_tokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 4, max = 4))
dtm_quadgram <- DocumentTermMatrix(clean_corpus, control = list(tokenize = quadgram_tokenizer))
n_docs = nDocs(dtm_quadgram)
dtm_quadgram <- removeSparseTerms(dtm_quadgram, (n_docs - 1.1) / n_docs)

tbl_dtm_quadgram <- as_tibble(as.matrix(dtm_quadgram))
tbl_frequencies <- summarise_all(tbl_dtm_quadgram, funs(sum))
quadgram_frequencies <- unlist(tbl_frequencies)

print(paste('The number of quadgrams with more than 2 occurance is', length(quadgram_frequencies[which(
```

```
## [1] "The number of quadgrams with more than 2 occurance is 2"
```

```
quadgram_frequencies[which(quadgram_frequencies > 2)]
```

```
## amazon services llc amazon     services llc amazon eu
##                          4                          4
```