Data Science Capstone – Milestone Report

Li Xu

Summary

In this report, we study the HC Corpora Data Set. We first read the data set and create a basic statistical summary about the number of lines and words. Then we clean this data set by removing numbers, whitespaces, punctuations and profanity words. We then establish the unigram, bigram, trigram, 4-gram models and list the top 20 grams.

Description of the Data Set

In the capstone project, we study the data set from a corpus called HC Corpora. You can find the description of this data set from the following website

```
http://www.corpora.heliohost.org/aboutcorpus.html
```

It can be downloaded from the following URL

```
https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip
```

After unzipping the file, we will find a folder containing four subfolders, each of which contains the blogs, news and twitters written in German, English, Finnish and Russian, respectively. The sizes of the files can be summarized as follows:

```
German:
     blogs     83 Megabytes
      news     93 Megabytes
  twitters     73 Megabytes
English:
     blogs    205 Megabytes
      news    200 Megabytes
  twitters    163 Megabytes
Finnish:
     blogs    105 Megabytes
      news     92 Megabytes
  twitters     24 Megabytes
Russian:
     blogs    114 Megabytes
      news    116 Megabytes
   twitter    102 Megabytes
```

Reading the Data Set and Cleaning

In this report, we study the blogs, news and twitters written in English.

By typing

```
iblogs<-file("final/en_US/en_US.blogs.txt")
blogs<-readLines(iblogs,encoding="UTF-8")
close(iblogs)
inews<-file("final/en_US/en_US.news.txt")
news<-readLines(inews,encoding="UTF-8")
close(inews)
itwitter<-file("final/en_US/en_US.twitter.txt")
twitters<-readLines(itwitter,encoding="UTF-8")
close(itwitter)
```

we read the data set including blogs, news and twitters into three vectors "blogs", "news", "twitters", respectively.

We count the number of lines and total number of words in each file by typing

```
wordsinblogs<-sapply(gregexpr("\\S+", blogs), length)
wordsinnews<-sapply(gregexpr("\\S+", news), length)
wordsintwitters<-sapply(gregexpr("\\S+", twitters), length)
lengths<-c(length(blogs),length(news),length(twitters))
sums<-c(sum(wordsinblogs),sum(wordsinnews),sum(wordsintwitters))
summarize<-rbind(lengths,sums)
colnames(summarize)<-c("blogs","news","twitters")
rownames(summarize)<-c("number of lines","number of words")
summarize
```

and obtain the following table

```
                   blogs    news twitters
number of lines   899288   77259  2360148
number of words 37334131 2643969 30373543
```

We then summarize the number of words in each line by typing

```
summary(wordsinblogs)
summary(wordsinnews)
summary(wordsintwitters)
```

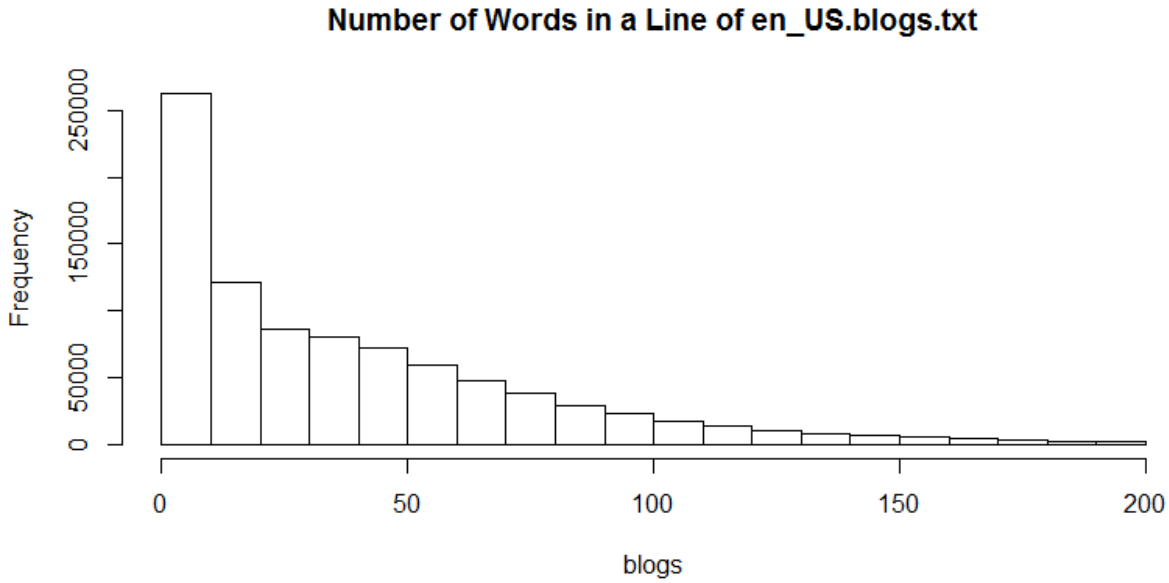
and obtain the following table

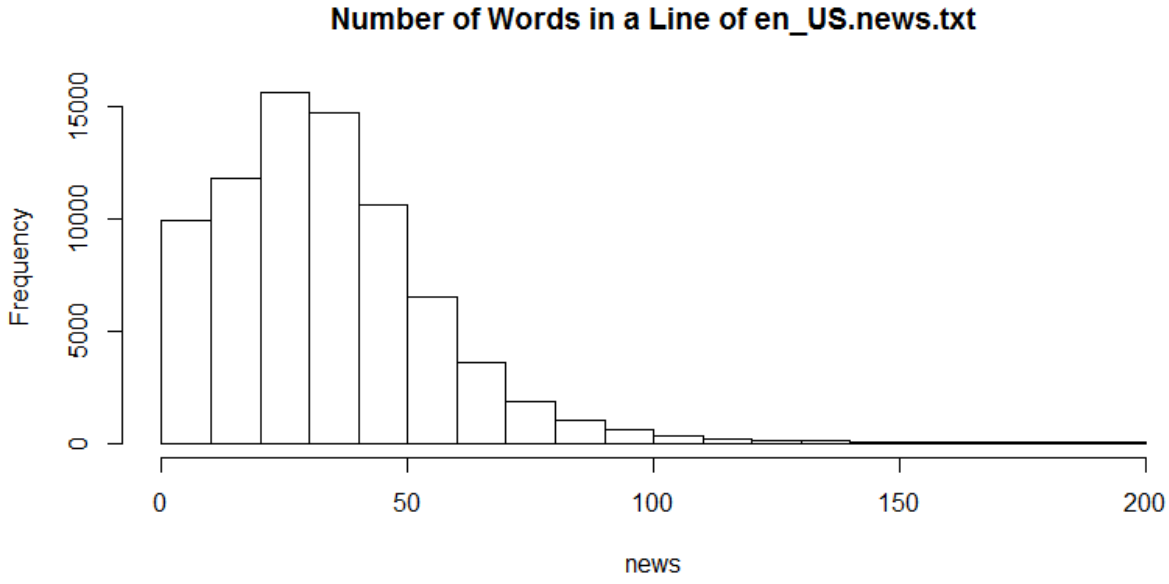
```
           Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   blogs   1.00    9.00   28.00   41.52   59.00 6630.00
    news   1.00   19.00   31.00   34.22   45.00 1031.00
twitters   1.00    7.00   12.00   12.87   18.00   47.00
```

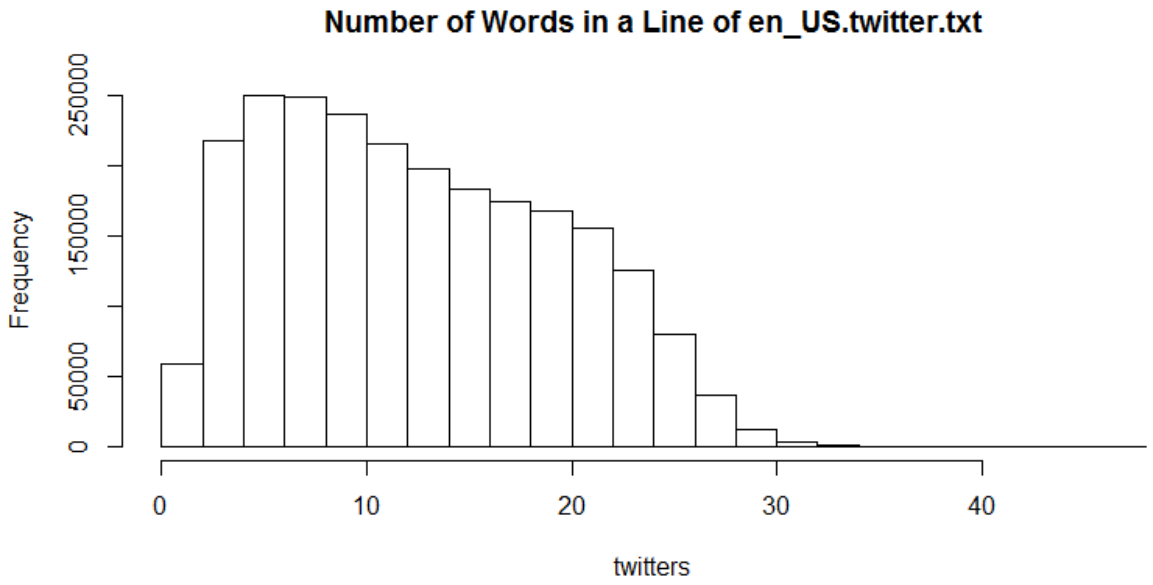
We then plot three histograms of the number of words in each line. Since most of the lines contain less than 200 words, we first . Then by typing

```
blogs<-wordsinblogs[wordsinblogs<200]
hist(blogs,main="Number of Words in a Line of en_US.blogs.txt")
news<-wordsinnews[wordsinnews<200]
hist(news,main="Number of Words in a Line of en_US.news.txt")
twitters<-wordsintwitters[wordsintwitters<200]
hist(twitters,main="Number of Words in a Line of en_US.twitter.txt")
```

we obtain the three histograms about the number of words in a line:







Since the number of lines in the data set is so huge, we randomly pick 10000 lines from the three files by typing

```
DataSet<-sample(paste(blogs,news,twitters),size=10000)
```

Then we remove numbers, whitespaces, punctuations, and convert all uppercase letters into lowercase for future convenience by typing

```
DataSet<-Corpus(VectorSource(DataSet))
DataSet<-tm_map(DataSet, removeNumbers)
DataSet<-tm_map(DataSet, stripWhitespace)
DataSet<-tm_map(DataSet, removePunctuation)
DataSet<-tm_map(DataSet, content_transformer(tolower))
```

A list of profanities can be downloaded from the following URL

```
https://github.com/quellhorst/negative-keywords/blob/master/profanity.txt
```

We then use this file to filter the profanities by typing

```
con<-file("profanity.txt")
Profanity<-readLines(con,encoding="UTF-8")
close(con)
ProfanityWords<-VectorSource(Profanity)
DataSet<-tm_map(DataSet, removeWords, ProfanityWords)
```

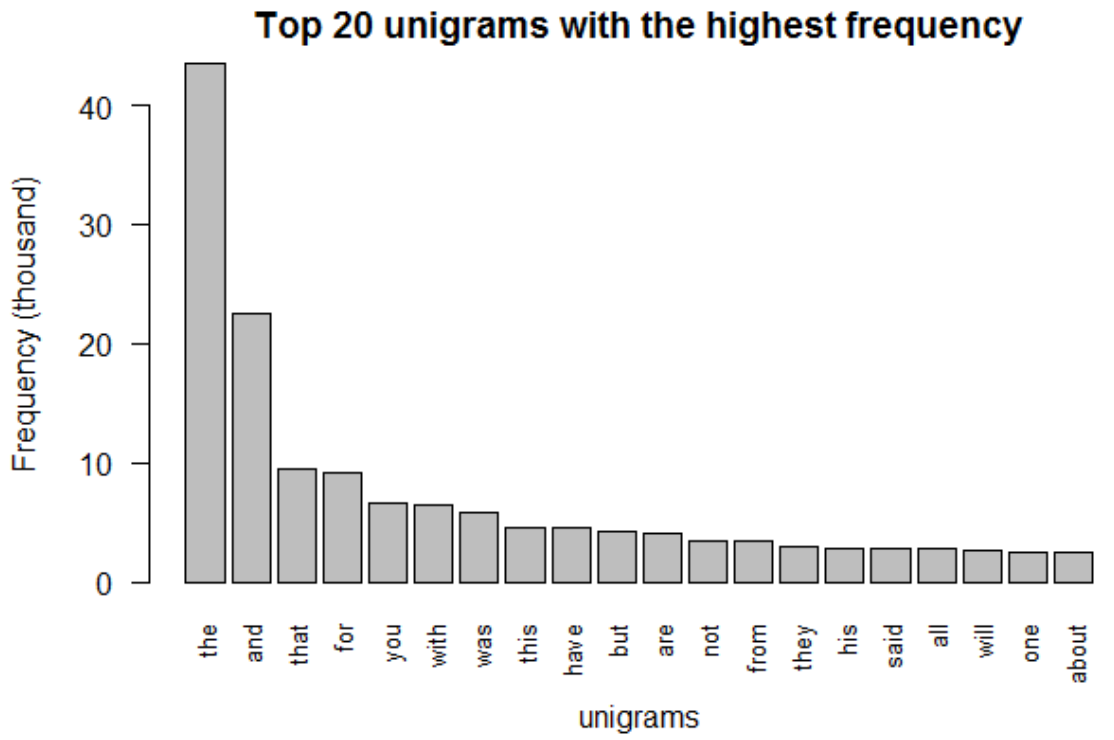
N-gram modeling

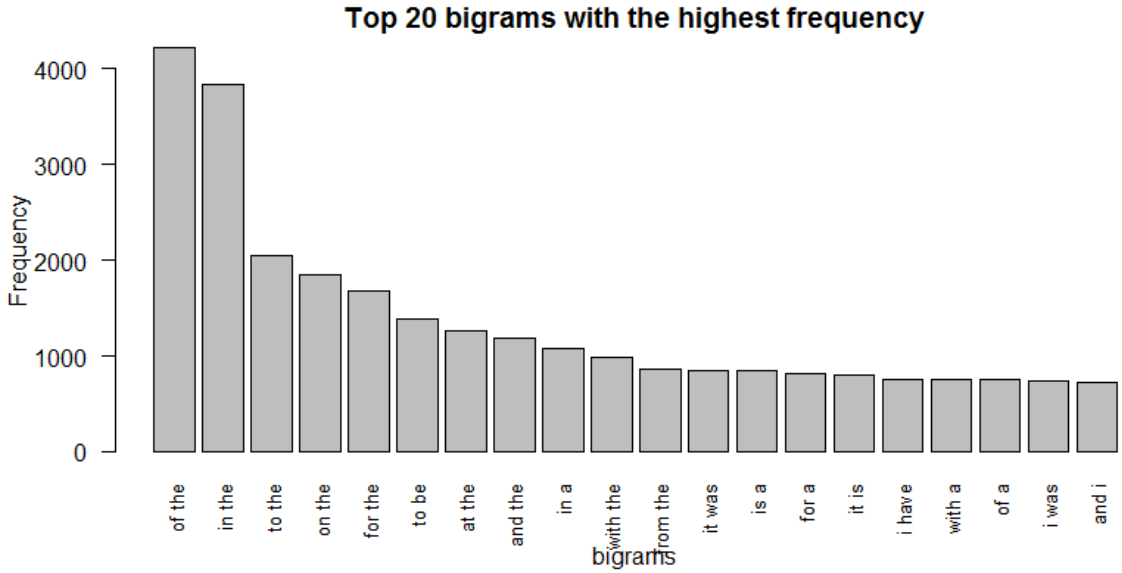
We establish unigram (1-gram), bigram (2-gram), trigram (3-gram) and 4-gram models for the data set by typing

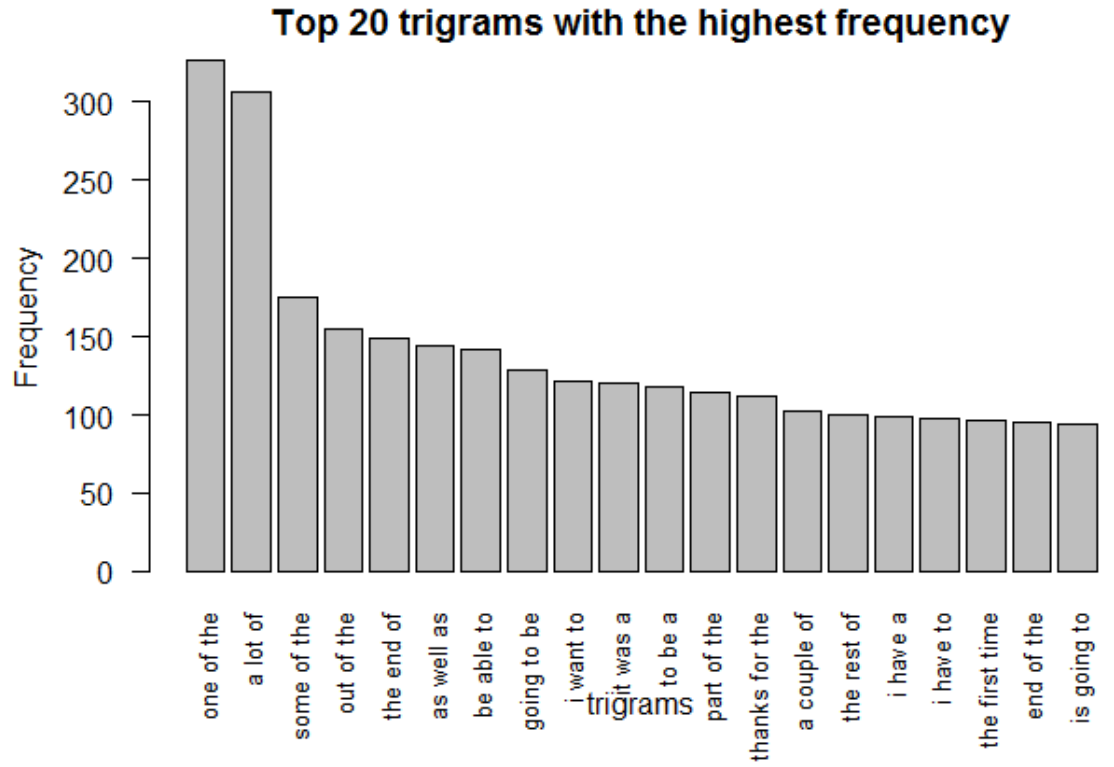
```
#unigram
unitoken<-function(x){
    NGramTokenizer(x,Weka_control(max=1,min=1))
}
unimatrix<-as.matrix(TermDocumentMatrix(DataSet, control=list(tokenize=unitoken)))
unirowsum<-rowSums(unimatrix)
unigram<-data.frame(unigram=names(unirowsum),freq=unirowsum)
unigramsorted<-unigram[order(-unigram$freq),]
par(mar = c(5, 5, 2, 2) + 0.2)
barplot(unigramsorted[1:20,]$freq/1000, horiz=F, cex.names=0.8, xlab="unigrams",
    ylab="Frequency (thousand)",las=2,names.arg=unigramsorted[1:20,]$unigram,
    main="Top 20 unigrams with the highest frequency")
#bigram
bitoken<-function(x){
    NGramTokenizer(x,Weka_control(max=2,min=2))
}
bimatrix<-TermDocumentMatrix(DataSet, control=list(tokenize=bitoken))
largefreq<-findFreqTerms(bimatrix,lowfreq=10)
birowsum<-rowSums(as.matrix(bimatrix[largefreq,]))
bigram<-data.frame(bigram=names(birowsum),freq=birowsum)
bigramsorted<-bigram[order(-bigram$freq),]
par(mar = c(5, 5, 2, 2) + 0.2)
barplot(bigramsorted[1:20,]$freq, horiz=F, cex.names=0.8, xlab="bigrams",
    ylab="Frequency",las=2,names.arg=bigramsorted[1:20,]$bigram,
    main="Top 20 bigrams with the highest frequency")
#trigram
tritoken<-function(x){
    NGramTokenizer(x,Weka_control(max=3,min=3))
}
trimatrix<-TermDocumentMatrix(DataSet, control=list(tokenize=tritoken))
largefreq<-findFreqTerms(trimatrix,lowfreq=10)
trirowsum<-rowSums(as.matrix(trimatrix[largefreq,]))
trigram<-data.frame(trigram=names(trirowsum),freq=trirowsum)
trigramsorted<-trigram[order(-trigram$freq),]
par(mar = c(7, 5, 2, 2) + 0.2)
barplot(trigramsorted[1:20,]$freq, horiz=F, cex.names=0.8, xlab="trigrams",
    ylab="Frequency",las=2,names.arg=trigramsorted[1:20,]$trigram,
    main="Top 20 trigrams with the highest frequency")
#four-gram
fourtoken<-function(x){
    NGramTokenizer(x,Weka_control(max=4,min=4))
}
fourmatrix<-TermDocumentMatrix(DataSet, control=list(tokenize=fourtoken))
largefreq<-findFreqTerms(fourmatrix,lowfreq=10)
fourrowsum<-rowSums(as.matrix(fourmatrix[largefreq,]))
fourgram<-data.frame(fourgram=names(fourrowsum),freq=fourrowsum)
fourgramsorted<-fourgram[order(-fourgram$freq),]
par(mar = c(5, 5, 2, 2) + 0.2)
barplot(fourgramsorted[1:20,]$freq, horiz=F, cex.names=0.8, xlab="",
    ylab="Frequency",las=2,names.arg=fourgramsorted[1:20,]$fourgram,
    main="Top 20 4-grams with the highest frequency")
```

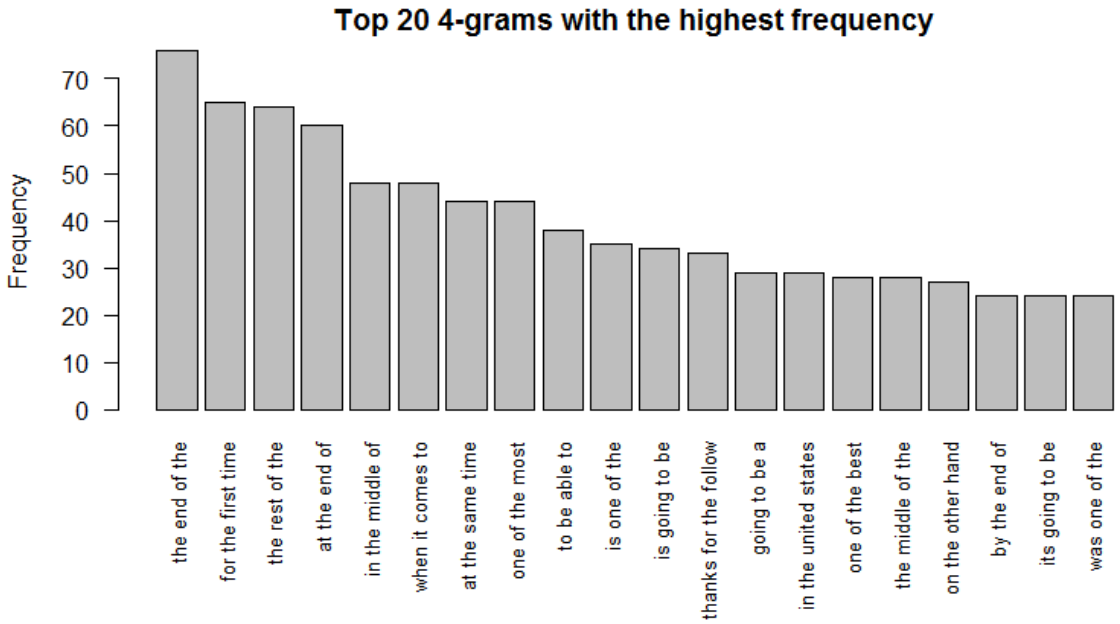
Then we obtain the top 20 unigrams, bigrams, trigrams and 4-grams with the highest frequency as the

following three figures.






Top 20 trigrams with the highest frequency
Frequency
0
50
100
150
200
250
300
one of the
a lot of
some of the
out of the
the end of
as well as
be able to
going to be
i want to
it was a
to be a
part of the
thanks for the
a couple of
the rest of
i have a
i have to
the first time
end of the
is going to
trigrams


Top 20 4-grams with the highest frequency
Frequency
0
10
20
30
40
50
60
70
the end of the
for the first time
the rest of the
at the end of
in the middle of
when it comes to
at the same time
one of the most
to be able to
is one of the
is going to be
thanks for the follow
going to be a
in the united states
one of the best
the middle of the
on the other hand
by the end of
its going to be
was one of the