Week 1

C. Kalinowski

3/1/2021

Task 0 - Understanding the Problem

Download the data

The data was downloaded and unzipped. It contains a folder named "final" with 4 subfolders.

- German (de_DE)
- English (en_US)
- Finnish (fi_FI)
- Russian (ru_RU)

Each folder contains 3 files beginning with the folder name and ending with the source name.

- blogs (.blogs.txt)
- news (.news.txt)
- twitter (.twitter.txt)

The path to Russian blogs is "../Data/final/ru_RU/ru_RU.blogs.txt"

Load the data into R

readLines outputs a character vector of length y, and you can access specific elements with the square brackets.

fileReader is a basic first pass at a function that takes a file name x and a length y. It opens a connection to file x for read purposes, inputs y lines. It then closes the connection and returns a character vector of length y.

```
sampleReader<-function(x,y=-1L,z="en_US"){
      ## Reads a specified number of lines from an input text from the data set
      ## First paste together file name from input x (either twitter, blogs, or news)
      ## Assume language is english, but can be switched to de, ru, or fi with z arg
      ## Open file connection, input y lines, or all if not specified
      ## close connection, return input lines as character vector of length y

      filename<-paste("../Data/final/",z,"/",z,".",x,".txt",sep="")
      con<-file(filename,"r")
      inputtext<-readLines(con,n=y)
      close(con)
      inputtext
}

myline<-sampleReader("twitter",2)
myline
```

```
[1] "How are you? Btw thanks for the RT. You gonna be in DC anytime soon? Love to see you. Been way, wa
[2] "When you meet someone special... you'll know. Your heart will beat more rapidly and you'll smile f
```

Task 1 - Getting and Cleaning the Data

Tokenization

Tokenization is splitting lines into words. A simple tokenizer, according to Jurafsky's lectures, splits on non-alpha characters.

A basic tokenizer is found in the tau package. It simply splits based on whitespace.

```
library(tau)
head(tokenize(myline))
```

```
[1] "How" " "   "are" " "   "you" "?"
```

A better option is probably the boost tokenizer in tm, which keeps punctuation...

```
library(tm)
```

```
Loading required package: NLP
```

```
head(Boost_tokenizer(myline))
```

```
[1] "How"    "are"    "you?"   "Btw"    "thanks" "for"
```

...or the MC tokenizer, which doesn't.

```
head(MC_tokenizer(myline))
```

```
[1] "How"    "are"    "you"    "Btw"    "thanks" "for"
```

Profanity Removal

Probably the best way to deal with this is to use an already-generated list of words like the google bad words list. There is a 2010 list available at "https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/badwordslist/badwords.txt", which was downloaded and stored as "../Data/badwords.txt"

The following "clean" tokenizer takes a character vector as input. It tokenizes it using the MC tokenizer, and then compares it against the bad words list. Anything in the tokenized vector that is on the list is replaced with the string "PRFNTY", and the "cleaned" vector is returned as output.

```
cleanToken<-function(x){
      ## creates "clean" set of tokens from input x
      ## uses MC_tokenizer to create token list
      ## checks token list against badwords list
      ## replace tokens on the bad list with "PRFNTY"
      ## return token list

      token<-MC_tokenizer(x)
      badWordsList<-readLines("../Data/badwords.txt",n=-1L,warn=FALSE)
      checkToken<-token %in% badWordsList
      token[checkToken]<-"PRFNTY"
      token
}

myclean1<-cleanToken(myline[1])
myclean1
```

```
 [1] "How"     "are"     "you"     "Btw"     "thanks"  "for"     "the"
 [8] "RT"      "You"     "gonna"   "be"      "in"      "DC"      "anytime"
```

```
[15] "soon"    "Love"    "to"      "see"     "you"     "Been"    "way"
[22] "way"     "too"     "long"
```