A quick review of corpus analyses we learnt about so far

Sowmya Vajjala

2/21/2018

Over the past 6 weeks, we discussed a range of corpus analysis methods in R (roughly covering Chapters 1–9 in the textbook). I started talking about text classification this week, so now is the time to look back on how much we already know. I tried to summarize everything about corpus analysis we learnt into a couple of functions, which you can use as is for doing further analysis of new texts.

(Things not included in the tutorial: processing different formats of files, writing R markdown etc. Only actual text analysis topics are covered here.)

Here are the functions: (you can see all of them in CorpusAnalysisFunctions.R file)

This function get_words_vector() below takes a file, and does the following:

- reads it into R using scan function
- makes a long string out of all lines of the text, and lower cases it
- removes punctuation
- converts this into a vector of words and gives it back to us.

```
get_words_vector <- function(file_path)
{
  fulltext <- scan(file_path, what = "character", sep = "\n")
  fulltext_as_string <- tolower(paste(fulltext, collapse = " "))
  words_vector <- unlist(strsplit(tolower(fulltext_as_string), "\\W+"))
  return (words_vector)
}
```

The next function is a simple function that returns the most frequent words in a text. However, it now directly works on the words vector that we get from the above function. i.e., once we read and process the file, we don't have to do that again and again. We can just give that as input to other functions.

```
get_freq_words <- function(wordsvector, n)
{
  sorted_freqs <- sort(table(wordsvector), decreasing = TRUE)
  return(sorted_freqs[1:n])
}
```

The next thing we did was to calculate lexical variety. We spoke about three measures: type token ratio, average word frequency, and percentage of hapax legomena i.e., words that appear only once.

```
get_lex_variety <- function(wordsvector)
{
  words_freqs <- table(wordsvector) #I don't need a sorted table here.
  ttr <- 100 * length(words_freqs)/sum(words_freqs)
  awf <- sum(words_freqs)/length(words_freqs)
  hapax <- 100*sum(words_freqs==1)/length(words_freqs)
  result = c(ttr,awf,hapax)
  names(result) = c("TypeTokenRatio","avgWordFreq","hapaxPercent")
  return(result)
}
```

We then spoke about a Key Word in Context (KWIC) search. The function below does that. It is very similar to what I showed in the previous class. Only change is, instead of having multiple print statements, I just store all of them in a string vector.

```
getKwic <- function(wordsvector, word, context)
{
  final_result = c()
  word_index <- which(wordsvector == word)
  #If the word is not found in this file: then, there is no point in going
  #further down into the for loop. It will also throw an error:
  #"Error in if (start < 1) { : missing value where TRUE/FALSE needed"
  if(length(word_index) > 0)
  {
    for(i in 1:length(word_index))
    {
      start <- word_index[i] - context
      if(start < 1)
      {
        start <- 1
      }
      end <- word_index[i] + context
      if(end >= length(wordsvector))
      {
        end <- length(wordsvector)
      }
      # cat(start, end) #This prints only positions, not actual words.
      kwic <- paste(wordsvector[start:end], collapse = " ")
      final_result <- c(final_result,kwic)#, sep=" "))
    }
  }
  #Since the word is not in the file, I am just printing that message and moving on.
  else{
    final_result <- c("Word is not found in this file.")
  }
  return(final_result)
}
```

We then spoke about doing ngram analysis, using the ngram library in R. Here is a function that returns you most frequent n-grams:

```
get_ngrams <- function(wordsvector,ngramsize,n_frequent)
{
  #ngrams library expects a string, so, first convert this vector to a string.
  text_string <- paste(wordsvector, collapse=" ")
  ngrams <- ngram(text_string,n=ngramsize)
  top10ngrams <- head(get.phrasetable(ngrams), n = n_frequent)
  return(top10ngrams)
}
```

i.e., once you have a vector of words, using this get_ngrams(wordsvector,3,10) gives you 10 most frequent ngrams in the text, and so on.

Apart from these various analyses, we spoke about plotting information from these corpus analyses. The following two functions plot a normal frequency plot, and a dispersion plot respectively.

```
get_line_plot <- function(wordsvector,n)
{
  words_i_need <- get_freq_words(wordsvector,n)
  plot(words_i_need)
  #change this to whichever kind of plot you want. Look at ?plot for options.
}

get_dispersion_plot <- function(wordsvector, word)
{
  progress <- seq(1:length(wordsvector))
  word_presence <- which(wordsvector == word)
  length(word_presence)
  word_progression <- rep(NA, length(progress))
  word_progression[word_presence] <- 1
  plot(word_progression, main="Dispersion plot for the given word in the given vector",
       xlab="position in text", ylab=word, type="h", ylim=c(0,1), yaxt = 'n')
}
#Read more on plot options to understand what all these type, ylim, yaxt etc mean
```

Usage of these functions

Okay, now that we have all these functions, how are we supposed to use them? I saved all these into one file, CorpusAnalysisFunctions.R. One starting point is to just load this file into your R console using source function:

```
source('~/Dropbox/ClassroomSlides-BothCourses/LING410X/22FEB2018Tutorial/CorpusAnalysisFunctions.R')
```

```
## Warning: package 'ngram' was built under R version 3.4.3
```

You should ofcourse change this path to yours! However, easier way is to just open that file in Rstudio, and click on 'source' on the top Right of the top Left panel. It takes care of the file path issues.

Next task is to set your working directory to where the texts you want to analyze are. You can do that by navigating in the bottom right panel, and setting the working directory there. Otherwise, you can get the full file path and just use that and not bother about setting the working directory. My working directory has about 10 text files I downloaded from Project Gutenberg. I will show some examples of using these functions on those files below:

Get Lexical Variety Measures for all files in my corpus

Let us say I want to get different lexical variety measures for all the files in my corpus folder, here is how to do it, using a for loop.

```
setwd("~/Dropbox/ClassroomSlides-BothCourses/LING410X/22FEB2018Tutorial/corpus")
file.names <- dir(getwd(), pattern = "\\.txt")
my_final_output <- c()
for (f in file.names) {
  x <- get_words_vector(f)
  my_final_output <- rbind(my_final_output, get_lex_variety(x))
}
```

Your final output contains one row per text, each row having three columns. Remember that get_lex_variety() function gives me three measures per text. So, if I just want to collect this output for all files to open in a

spreadsheet later, I can use write.csv() which we used before.

```
write.csv(my_final_output,"temp.csv")
```

I can even define one more function that does all this.

Get the most frequent n-grams in a given text

We can use some of these functions to get our most frequent ngrams in a given text.

```
setwd("~/Dropbox/ClassroomSlides-BothCourses/LING410X/22FEB2018Tutorial/corpus")
wordsvector <- get_words_vector("text1.txt")
ngrams <- get_ngrams(wordsvector, 3, 20)
#This gives me 20 most frequent trigrams in this text.
#I am printing all output, you can change this to print whatever you want.
```

So, you can use these functions on your R console as if they are built in R functions, once you load the R file using the source() function in the beginning. Similarly, you can use these functions for other purposes.

Exercises:

1. Plot the 15 most frequent words in text5.txt
2. How many common 3grams exist between text5.txt and text6.txt, if I take, say, 50 most frequent 3grams in each? Is the number of common ngrams going to be more or less if I take unigrams instead of 3grams? Think about it, and then see if the results reflect your thinking.
3. Store the keyword in context function output into a csv file, open it in Excel, and try splitting the string column into word columns by choosing a white space as a column separator.
4. Get the dispersion plots for “stockmann” and “petra” in text1.txt
5. Can you make these plots appear side by side?