Assignment-3: Solution Discussion

Sowmya Vajjala

2/26/2018

Here is one solution for your Assignment 3. I am using the CorpusAnalysisFunctions.R file and the functions I defined there, instead of typing everything again.

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

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

Question 1

Read the following URL's content into R and calculate the following: http://www.gutenberg.org/cache/epub/2446/pg2446.txt - lexical variety in that article (unique words/total words). Consider only words - not punctuation markers and other such non-word characters. - 10 most frequent words in the article Now, write a short 1 page report on the article you picked, and the results you got, and how you got them (i.e., R commands you used)

Regarding reading the file: There are two ways to do this: I can download and store on my computer and use scan function, or I can directly read them using scan function and giving the url instead.

```
url <- "http://www.gutenberg.org/cache/epub/2446/pg2446.txt"
words_vec <- get_words_vector(url)
lex_var <- get_lex_variety(words_vec)
freq10words <- get_freq_words(words_vec,10)
lex_var
```

```
## TypeTokenRatio    avgWordFreq   hapaxPercent
##      10.351568       9.660372      50.985222
```

```
freq10words
```

```
## wordsvector
##       the        to       you         i stockmann        of       and
##      1574       962       953       883       857       804       731
##         a      that        is
##       701       628       626
```

Question 2

Download the content of any 2 books by one author from gutenberg.org (in .txt format!). In R, read the

les, remove punctuations, lowercase all words, split the text into words based on white space seperation, and make a list of unique unigrams, 2 and 3 grams in the book. Compare the lists for both books thus created, and create a list of unigrams, bi and trigrams that appear in both texts. Each time, print the following to console: Number of n (n=1,2,3) grams in Book1, in Book 2, and their intersection. Submit your assignment as a R-markdown submission. Let us say I take these two books: http://www.gutenberg.org/cache/epub/2446/pg2446.txt

http://www.gutenberg.org/cache/epub/7172/pg7172.txt

-There are two ways to do this: I can download these and store on my computer, or I can directly read them using scan function.

```
ibsen1_wordsvec <- get_words_vector("http://www.gutenberg.org/cache/epub/2446/pg2446.txt")
ibsen2_wordsvec <- get_words_vector("http://www.gutenberg.org/cache/epub/7172/pg7172.txt")
```

So, here is the issue now: in my CorpusAnalysisFunctions.R, the get_ngrams function takes three arguments - words vector, ngram size, n_frequent. So, we need to specify the number of frequent ngrams we want. But, in my question, I asked for all unique ngrams. So, what should I do?

Here are two possible solutions: 1) You can do how we did in the class earlier (in Week 6 - 13th and 15th Feb classes) - convert ibsen1_wordsvec1, ibsen1_wordsvec2 into strings instead of vectors (Why? ngram function expects strings, not vectors)

```
ibsen1_string <- paste(ibsen1_wordsvec, collapse=" ")
ibsen2_string <- paste(ibsen2_wordsvec, collapse=" ")

unigrams_ibsen1 <- ngram(ibsen1_string,n=1)
bigrams_ibsen1 <- ngram(ibsen1_string,n=2)
trigrams_ibsen1 <- ngram(ibsen1_string,n=3)

unigrams_ibsen2 <- ngram(ibsen2_string,n=1)
bigrams_ibsen2 <- ngram(ibsen2_string,n=2)
trigrams_ibsen2 <- ngram(ibsen2_string,n=3)

#Get a table with ngrams, frequencies etc from the above ones:
unigrams_vector_ibsen1 <- get.ngrams(unigrams_ibsen1)
bigrams_vector_ibsen1 <- get.ngrams(bigrams_ibsen1)
trigrams_vector_ibsen1 <- get.ngrams(trigrams_ibsen1)

unigrams_vector_ibsen2 <- get.ngrams(unigrams_ibsen2)
bigrams_vector_ibsen2 <- get.ngrams(bigrams_ibsen2)
trigrams_vector_ibsen2 <- get.ngrams(trigrams_ibsen2)

#Number of unique uni, bi, tri grams in Ibsen 1
length(unigrams_vector_ibsen1)
```

```
## [1] 3654
```

```
length(bigrams_vector_ibsen1)
```

```
## [1] 18327
```

```
length(trigrams_vector_ibsen1)
```

```
## [1] 29469
```

```
#Number of unique uni, bi, tri grams in Ibsen 2
length(unigrams_vector_ibsen2)
```

```
## [1] 5578
```

```
length(bigrams_vector_ibsen2)
```

```
## [1] 34537
```

```
length(trigrams_vector_ibsen2)
```

```
## [1] 53820
```

```
#Intersection:
intersectuni <- intersect(unigrams_vector_ibsen1, unigrams_vector_ibsen2)
intersectbi <- intersect(bigrams_vector_ibsen1, bigrams_vector_ibsen2)
intersecttri <- intersect(trigrams_vector_ibsen1, trigrams_vector_ibsen2)

#Number of intersecting uni-, bi-, trigrams between the books.
length(intersectuni)
```

```
## [1] 2068
```

```
length(intersectbi)
```

```
## [1] 5232
```

```
length(intersecttri)
```

```
## [1] 3852
```

- However, clearly, we are copy pasting a lot of lines again and again. How about writing a function, which we can reuse?

In the below piece of R code, I am creating a new function, that takes a file path as input, and gives back to me a vector, where first item is a vector of unique unigrams, second item is a vector of unique bigrams, third item is a vector of unique trigrams

```
a3q2_function <- function(filepath)
{
  fulltext <- scan(filepath, what = "character", sep = "\n")
  fulltext_as_string <- paste(fulltext, collapse = " ")
  words_vector <- unlist(strsplit(tolower(fulltext_as_string), "\\W+"))
  fulltext_as_string <- paste(words_vector, collapse = " ")
  all_unigrams_in_this_text <- ngram(fulltext_as_string,n=1)
  all_bigrams_in_this_text <- ngram(fulltext_as_string,n=2)
  all_trigrams_in_this_text <- ngram(fulltext_as_string,n=3)
  unique_unigrams_vector <- get.ngrams(all_unigrams_in_this_text)
  unique_bigrams_vector <- get.ngrams(all_bigrams_in_this_text)
  unique_trigrams_vector <- get.ngrams(all_trigrams_in_this_text)
  ngrams_vectors <- list(unique_unigrams_vector,unique_bigrams_vector,unique_trigrams_vector)
  return(ngrams_vectors)
}
```

Let us see what happens if I just use this function:

```
ibsen1_ngrams_vectors <- a3q2_function("http://www.gutenberg.org/cache/epub/2446/pg2446.txt")
ibsen2_ngrams_vectors <- a3q2_function("http://www.gutenberg.org/cache/epub/7172/pg7172.txt")

#To get length of unigrams in both texts
length(ibsen1_ngrams_vectors[[1]])
```

```
## [1] 3654
```

```
length(ibsen2_ngrams_vectors[[1]])
```

```
## [1] 5578
```

```
#To get length of bigrams in both texts
length(ibsen1_ngrams_vectors[[2]])
```

```
## [1] 18327
```

```
length(ibsen2_ngrams_vectors[[2]])
```

```
## [1] 34537
```

```
#To get length of trgrams in both texts
length(ibsen1_ngrams_vectors[[3]])
```

```
## [1] 29469
```

```
length(ibsen2_ngrams_vectors[[3]])
```

```
## [1] 53820
```

```
#To get number of intersecting ngrams in both texts:
length(intersect(ibsen1_ngrams_vectors[[1]],ibsen2_ngrams_vectors[[1]]))
```

```
## [1] 2068
```

```
length(intersect(ibsen1_ngrams_vectors[[2]],ibsen2_ngrams_vectors[[2]]))
```

```
## [1] 5232
```

```
length(intersect(ibsen1_ngrams_vectors[[3]],ibsen2_ngrams_vectors[[3]]))
```

```
## [1] 3852
```