Introduction to basic Text Mining in R.

As published in Benchmarks RSS Matters, January 2014

http://web3.unt.edu/benchmarks/issues/2014/01/rss-matters

Jon Starkweather, PhD
RSS hosts a number of “Short Courses”.
A list of them is available at:
http://www.unt.edu/rss/Instructional.htm

Those interested in learning more about R, or how to use it, can find information here:
http://www.unt.edu/rss/class/Jon/R_SC
Introduction to basic Text Mining in R.

This month, we turn our attention to text mining. Text mining refers to the process of parsing a selection or corpus of text in order to identify certain aspects, such as the most frequently occurring word or phrase. In this simple example, we will (of course) be using R to collect a sample of text and conduct some rudimentary analysis of it. Keep in mind, this article simply provides a cursory introduction to some text mining functions.

First, we need to retrieve or import some text. We will use the University of North Texas (UNT) policy which governs Research and Statistical Support (RSS) services; UNT policy 3 - 5 for this example. We can use the ‘readLines’ function available in the ‘base’ package to retrieve the policy from the UNT Policy web site. Notice this policy’s HTML page is 305 lines long, which includes all the HTML formatting; not just the text of the policy.

Next, we need to isolate the actual text of the policy’s HTML page. This can take some investigating – using the head and tail functions or simply pasting the HTML page into a text editor will allow us to identify the line number(s) which contain the actual text of interest. Once identified, we can use a ‘which’ function to isolate or extract the lines we are interested in parsing. We notice below the actual text of the policy exists on lines 192 through 197, prefaced by the “Total University” header on line 189. We use the ‘which’ function to identify the line (189) with the header statement, then add 3 to it to arrive at line 192 (id.1; which identifies the first line of the policy). Then, we add a further 5 to that (192 + 5 = 197) to identify the last line of the policy (id.2). Then, we create a new object (‘text.data’) which contains only those lines which contain the text of the policy.

[http://cran.r-project.org/]
Now we are left with a vector object (text.data), which contains only the 6 lines of text of the policy (i.e. each paragraph of the policy has become a character string line of the vector).

Next, we need to remove the HTML tags (e.g. `<p>`) from each line of text. Generally multiple characters can be given in the ‘pattern’ argument within one implementation of ‘gsub’ function; but here we are using two implementations so that we can specifically remove each of the HTML (paragraph) tags while leaving in place all other instances of the letter ‘p’. Notice, we are using the ‘replacement’ argument to eliminate any instance of the ‘pattern’ argument – using nothing between the quotation marks of the ‘replacement’ argument.

Now we can load the ‘tm’ package (Feinerer & Hornik, 2013) and convert our vector of character strings into a recognizable corpus of text using the ‘VectorSource’ function and the ‘Corpus’ function.
Next, we make some adjustments to the text; making everything lower case, removing punctuation, removing numbers, and removing common English stop words. The `tm_map` function allows us to apply transformation functions to a corpus.

Next we perform stemming, which truncates words (e.g., “compute”, “computes” & “computing” all become “compt”). However, we need to load the ‘SnowballC’ package (Bouchet-Valat, 2013) which allows us to identify specific stem elements using the ‘tm_map’ function of the ‘tm’ package.
Next, we remove all the empty spaces generated by isolating the word stems in the previous step. We use the `stripWhitespace` argument of the `tm_map` function to accomplish this task.

Now we can actually begin to analyze the text. First, we create something called a Term Document Ma-
trix (TDM) which is a matrix of frequency counts for each word used in the corpus. Below we only show the first 20 words and their frequencies in each document (i.e. for us, each ‘document’ is a paragraph in the original policy).

Next, we can begin to explore the TDM, using the ‘findFreqTerms’ function, to find which words were used most. Below we specify that we want term / word stems which were used 8 or more times (in all documents / paragraphs).

Next, we can use the ‘findAssocs’ function to find words which associate together. Here, we are specifying the TDM to use, the term we want to find associates for, and the lowest acceptable correlation limit with that term. This returns a vector of terms which are associated with ‘comput’ at \( r = 0.60 \) or more (correlation) – and reports each association in descending order.
If desired, terms which occur very infrequently (i.e., sparse terms) can be removed; leaving only the ‘common’ terms. Below, the ‘sparse’ argument refers to the MAXIMUM sparse-ness allowed for a term to be in the returned matrix; in other words, the larger the percentage, the more terms will be retained (the smaller the percentage, the fewer [but more common] terms will be retained).

We can review the terms returned from a specific sparse-ness by using the ‘inspect’ function with the TDMs containing those specific sparse-ness rates (i.e., the terms retained at specific spare-ness levels). Below we see the 22 terms returned when sparse-ness is set to 0.60.
Next, we see the 5 terms returned when sparse-ness is set to 0.20 – fewer terms which occur more frequently (than above).

Conclusions

As stated in the introduction to this article, the above functions provide only a cursory introduction to importing some text and parsing it. Those seeking more information may want to consider taking a look at the ‘Natural Language Processing’ Task View at CRAN (Fridolin, 2013; link provided below).
task view provides information on a number of packages and functions available for processing textual data, including an R-Commander plugin which new R users are likely to find easier to use (at first).

For more information on what R can do, please visit the Research and Statistical Support Do-It-Yourself Introduction to R course website. An Adobe.pdf version of this article can be found here.

Until next time; “no twerking while working!”

References & Resources


Tutorial on text mining (author name not stated on page): http://www.exegetic.biz/blog/2013/09/text-mining-the-complete-works-of-william

This article was last updated on January 6, 2014.

This document was created using \LaTeX