Keyword Analysis using R

Dr. Lei Guo, Ankit Sharma (Boston University)

11/1/2015

1 Abstract

The code for Keyword Analysis was written by me when working as a Research Assistant for Prof. Lei Guo at the College of Communication at Boston University. I was learning R on the fly and writing the code for keyword analysis so you can probably optimize it more. I have included a table which gives the run times for different data sizes.
The data analysis was performed on data collected from twitter for US presidential elections 2012. This document explains how to use the code for Exact, Partial and Not matches for keyword analysis. Kindly pay attention to the way you should arrange your input files and naming of columns.

2 Input Files

The input files are .CSV extension which means (if you don't already know) that the delimiter is Comma(,). The default format for Excel is .XLSX and for LibreOffice is .ODF so you have to specify that you want to save the file as .CSV.
There are three separate input files for Exact Match, Partial Match, and Not Match. The column names should be the same for keywords belonging the same class. For example - suppose you have a class named 'Unemployment' which contains keywords for exact, partial and not match. For simplicity, let us see an example using the table below.

Table 1: A sample keyword column

Unemployment
job
lay off
employment
steve jobs

This table shows an example column with the first two words being Exact match, next one is partial match and the last one is not match. We need to create three input files. One each for Exact, Partial and not match. The tables below show the three input columns after the split.

Table 2: Exact Match

Unemployment Exact
job
lay off

Table 3: Partial Match

Unemployment Partial
employment

Table 4: Not match

Unemployment Not match
steve jobs

Remember to save your input files as .CSV extension. After this, your input files are ready and now on to some data analysis.

3 Data Analysis

3.1 Important Things to Note

- R only uses a single core of your CPU no matter how many you have.
- R stores all data into your physical memory by default. So you're constrained by the amount of RAM you have in your system.
- All the data analysis I did was on Windows 10, Intel i7-4790k Haswell processor, 16 GB RAM system using RStudio.

3.2 The Code

The Code is divided into three sections for Exact, Partial and Not matches. It first reads the input .CSV file and saves it as a dataframe. Since a tweet contains many unnecessary items like emojis, hyperlinks and other words which are not useful for data analysis so I first remove all the things we don't need. I used the Tmap package to remove punctuation, numbers and stopwords. I defined my own stopwords using the vector 'moreStopwords'. You can modify that vector according to your need.

Tmap converts your dataframe to a corpus for data analysis. Sometimes you may have a keyword which contains more than one word, like the one we saw in our example above ('steve jobs'). For such keywords, I used the RWeka package which has NGramTokenizer function which can match multiple word keyword. I declare the function 'myTokenizer' which calls the NGramTokenizer. The 'min' and 'max' parameters are for the length of minimum and maximum words in a keyword. For the input files I had, the minimum was 1 word and maximum was 4 words. You should change these parameters according to your keywords. I tested it with different length keywords and it works.

For exact match, we give one column of keywords using a for loop which loops over all the columns of keywords. The Document Term matrix has all the matches which we sum to get the number of matches for each class using rowsums and save the corresponding column in 'outFrame'. The column names are defined in cols and set as column names for 'outFrame'. Similar procedure for Partial and Not Match.

Note: The Column names defined in the cols vector should be same for all matches in their respective functions. Example, you

should have the name 'Unemployment' if that class has keywords for all the three matches. Look at the code if any lingering doubts.

The final step is to combine all the three dataframes and subtract the values for not matches. Then to get the binary matches of 0's and 1's, we run a for loop over all the values with a conditional.

And that's it, I now write the resulting dataframe to a CSV file. I tested my output with the training file I had and got the same results. Hopefully, it should work for you too.

3.3 Results

Here I include the times it took for data analysis for different data sizes I tested. I tested the times for all matches separately.

Table 5: Time taken for Data Analysis

Match	1000 tweets	10000 tweets	100000 tweets
Exact	1 minute	4 minuets	25 minutes
Partial	1 minute	2 minutes	22 minutes
Not Match	20 seconds	1 minute	6 minutes

The time for Not match is less because I had only one Not Match keyword. If you have more keywords, then the time will increase likewise. To get an idea, the Exact Match Keywords is a 57X17 dimension dataframe.

4 Resources

Here I list some of the resources I found useful. Since I was learning R on the fly, I'll also include resources for learning R.

- RStudio: *https://www.rstudio.com/*
- Learning R: *https://www.datacamp.com/*
- When in doubt, use this: *http://stackoverflow.com/*
- TMap package: *https://cran.r-project.org/web/packages/tmap/tmap.pdf*
- RWeka package: *https://cran.r-project.org/web/packages/RWeka/RWeka.pdf*