Evan Batzer

STA 141C

HW2

1/21/2019

**1. Write a function to compute the top n = 25 most heavily weighted words for each agency. Describe your approach to text preprocessing. What steps did you take in what order? Why?**

Steps:

1. Convert ASCII/Unicode

To begin, I thought it was most appropriate to standardize the format of character inputs for text processing. This involved converting non-UTF-8 inputs to UTF-8 to avoid any odd representation of certain characters that might cause issues with text processing.

1. To lowercase

I then converted all characters to their lowercase form before any sort of removal of stopwords and punctuation. Based on my subsequent consideration of function outputs, it seems that some functions in the *tm* package only operate on lowercase characters.

1. Remove punctutation

Then, I removed punctuation to facilitate removal of stopwords and stemming. I assumed this could help with odd cases where some words had odd punctuation, or in cases where punctuation at the end or beginning of certain words could change behavior of stemming and stopword removal functions – e.g. “running.” may not be appropriately trim to “run” because of the “.” character.

1. Stem words

To ease comparison across words with different suffixes, I then trimmed words down to their core “stems” – e.g., “running” and “runner” to “run”. This should allow our output word frequency tables to accurately capture the subject material of spending descriptions, rather than tabulating all unique words with shared stems.

1. Remove stopwords

Then, to downweight the importance of words such as “and” and “the”, which provide little meaning to an spending description, I removed stopwords using the removeWords(x, stopwords(“en”)) function provided in the *tm* package.

**2. Use this function to process the 91 agencies that have file sizes between 1 MB and 50 MB. (Recall 1 MB = 2ˆ20 = 1048576 bytes) Examine your result and use it to improve the function you wrote above. What did you have to change as you looked at the result?**

Initially, I had some issues with removing stopwords – my function only seemed to work properly when strings were first converted to lowercase. I also had issues with strsplit() picking up “” as a separate string that was then tabulated. A convenient workaround was to use another library, *tidytext*, that syncs well with *tm*, taking the sparse matrix generated from a *tm* corpus and producing a matrix of terms appearing in each string.

Source: https://cran.r-project.org/web/packages/tidytext/vignettes/tidying\_casting.html

I also noticed a number of terms that I was concerned were the result of some error in my function, such as "igfotigf", "igfcligf", and "igfctigf". After visiting the Federal Procurement Data System page, however, I learned that these special abbreviations refer to Inherently Governmental Functions of various types - "other functions", "critical functions", "and closely associated functions". However, these are not issues with my function, necessarily, and just unique terms that have meaning in the dataset.

Source: https://www.fpds.gov/fpdsng\_cms/index.php/en/newsroom/108-nherently-governmental-functions.html

**3. See table below**

**4. For the agencies listed above, are the results consistent with the name of the agency?**

**For example, do the words associated with the National Science Foundation seem related to science? Did you notice any strange results in these or in any other agencies?**

In general, the results seem to be surprisingly consistent with the name of the agency. The National Science Foundation appears to spend a large amount of its funds on research programs and operations, while border security is focused on border service, support, and maintenance, and the forest service pays primarily for helicopters and I assume a 737 airplane from Boise, ID. The FBI, interestingly, spends most of its funding on "inherently governmental functions", which may or may not be a deliberate attempt at concealing spending on their part.

**5. Make your program parallel on your local machine with parallel::clusterLapply and with parallel::clusterLapplyLB. Use 2 or more processes on your local machine and time both results. What do these functions do? Are they faster than the serial version of lapply? Which is fastest?**

Making the program parallel provided substantial reductions in the amount of time needed to call my text processing function on all relevant datasets. Both parLapply() and clusterApplyLB() completed the task in roughly 15 and 14 minutes, respectively, from around 45 minutes when running the function serially.

|  |  |  |
| --- | --- | --- |
| parLapply() | clusterApplyLB() | Lapply() |
| 904.18 | 817.80 | 2647.35 |

*Table 1: System time used to perform the textprocess() function across US spending datasets between 1MB and 50MB in size. Cluster utilized 4 processes on local machine.*

Reductions in speed did not follow a linear relationship with the number of clusters used, e.g. 4 clusters did not cause a ¼ reduction in the time to complete the task, likely due to processing power devoted to coordinating processes within the cluster and overall restrictions in processing speed and memory.

Both parLapply() and clusterApplyLB() separate tasks to different nodes within the cluster. parLapply() does so by separating the list of *n* arguments into *p* groups, with each of the *p* nodes in the cluster responsible for *n/p* function runs. The clusterApplyLB() function, a “load-balanced” version of clusterApply, first allocates *p* of the *n* elements in our vector of csv names to the nodes, then assigning new elements when a node completes a run. This is likely more efficient, particularly in our case where file sizes may vary considerably. Both are faster than the serial version of apply, with clusterApplyLB() being slightly faster than parLapply(). The benefit of load balancing will likely be more efficient if the processing costs of each element in the list of file names was more uneven.

**3. Show your results in a table for the following agencies:**

**• National Science Foundation (655)**

**• Federal Bureau of Investigation (262)**

**• U.S. Customs and Border Protection (778)**

**• Forest Service (110)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **NSF** | | **FBI** | | **C. and B. Protection** | | **Forest Service** | |
|  | word | weight | word | weight | word | weight | word | weight |
| 1 | research | 8944416167.56 | igfotigf | 6821816972.06 | servic | 8185627417.46 | helicopt | 24947408.34 |
| 2 | oper | 5808473559.01 | servic | 4333592367.56 | support | 7650501065.69 | servic | 24475751.72 |
| 3 | program | 4745588691.12 | support | 2555525629.30 | mainten | 5912325225.41 | type | 18738365.63 |
| 4 | scienc | 3777403094.67 | year | 1414078379.62 | igfotigf | 4322819103.78 | call | 16357868.90 |
| 5 | collabor | 3426075933.32 | option | 1253439031.75 | order | 4031272973.60 | need | 16357868.90 |
| 6 | support | 3332588913.60 | igfctigf | 974743006.91 | task | 3390677784.90 | 737boiseid | 13407052.81 |
| 7 | center | 3290007079.39 | mainten | 925793893.67 | border | 3099569512.46 | usfsbo | 13407052.81 |
| 8 | nation | 3188454264.51 | equip | 763542588.37 | system | 3031500077.28 | cwn | 11383000.34 |
| 9 | manag | 3020834396.12 | fbi | 759069883.71 | program | 2750837927.76 | public | 7343841.93 |
| 10 | mainten | 2379377920.27 | softwar | 754477306.85 | oper | 2521390271.30 | land | 7317568.93 |
| 11 | observatori | 2354985428.50 | fund | 659919659.36 | softwar | 2295948924.38 | use | 7303141.93 |
| 12 | system | 2206970410.64 | task | 638700173.49 | modern | 1938880156.55 | protect | 7300641.93 |
| 13 | servic | 2199140414.05 | order | 616754332.38 | manag | 1883461533.66 | administr | 6702567.98 |
| 14 | integr | 1807848728.48 | system | 613624749.25 | contract | 1750547426.87 | exclus | 5887919.13 |
| 15 | engin | 1783061878.68 | contract | 612974767.64 | logist | 1665648134.33 | forest | 4150985.03 |
| 16 | ocean | 1677326730.60 | igfcligf | 589711287.13 | applic | 1599704844.97 | attack | 3973640.44 |
| 17 | state | 1616009520.94 | function | 575698873.20 | aircraft | 1579830611.22 | wildfir | 3973640.44 |
| 18 | comput | 1466658360.35 | exercis | 566575969.71 | igfcligf | 1564155184.06 | usda | 3925075.44 |
| 19 | graduat | 1398574101.02 | develop | 535556521.07 | equip | 1407364171.71 | initi | 3689378.68 |
| 20 | fellowship | 1371300263.53 | hardwar | 510251409.19 | cbp | 1378431058.47 | year | 3375566.49 |
| 21 | antarct | 1356441465.96 | data | 493393875.50 | develop | 1373864795.09 | att | 2934640.29 |
| 22 | unit | 1345471704.45 | 0200 | 468005491.42 | secur | 1177870602.59 | circuit | 2934640.29 |
| 23 | advanc | 1317917277.33 | manag | 466119089.63 | offic | 1168557357.55 | calendar | 2927706.66 |
| 24 | facil | 1314623577.21 | program | 432746829.34 | igfctigf | 1136906442.97 | report | 2838410.27 |
| 25 | atmospher | 1273780280.23 | new | 425223581.58 | award | 1106567533.44 | fy05 | 2696275.60 |

Table 2: Term frequencies and weight, ranked from greatest to least weight. Datasets include “National Science Foundation”, “Federal Bureau of Investigation”, “Customs and Border Protection”, and “US Forest Service”.