Tracking New Abbreviations (newAbbrvReport)

Mark White | markhwhiteii@gmail.com

5/16/2017

This document describes an R function that will generate reports of abbreviations (i.e., words 2 or 3 letters in length) that have been used in chief complaint or admit reason fields for a current month but have *not* been used in previous months. As well as outputting the abbreviations, this function also generates the BioSense ID as well as the raw text in which the abbreviation appeared.

To use this function, you will need to first run the code in the `newAbbrvReport.R` document. (*Note*: For those interested in editing the code, I discuss it line-by-line in the `newAbbrvReportExplainer.pdf` document).

Preparing to use the `newAbbrvReport` function

First, you will have to load in the old data that to which you will be comparing the new month's data. This is located in the same file in which the `newAbbrvReport.R` document is stored, and this data file is named `abbrvCounts.csv`. It is *VERY* important that you load these data to an object named `oldcounts`. If you do not do this (which is case-sensitive), the code will not execute. You load this file with the simple `read.csv()` command:

```
oldcounts <- read.csv("abbrvCounts.csv", stringsAsFactors=FALSE)
```

Where `"abbrvCounts.csv"` is the directory in which the comparison data are stored. We would like to load the words as characters, not factors, so I indicated `stringsAsFactors=FALSE`.

After this, you will have to set up a connection using the `odbcConnect` function in the `RODBC` package. Load this package using the `library(RODBC)` command. An example of this code looks like:

```
channel <- odbcConnect("Biosense_Platform", "BIOSENSE\\username", "password")
```

Lastly, you will have to install the `dplyr`, `tidytext`, `stringr`, and `stringi` packages before using these functions, since the functions discussed here depend on those packages (as well as the `RODBC` package). After you have this set up, you are ready to run the functions.

The `newAbbrvReport` function

This function will return two data frames: first, a report of new abbreviations used in the specified month, along with context and Biosense IDs; second, an updated version of the `abbrvCounts` data file that now contains the specified month's words and counts.

This function has three arguments:
1. channel = the channel you set up through `odbcConnect`.
2. month = the month you want the report for (i.e., `"01"`, `"02"`, ..., `"12"`).
3. year = the year you want the report for (e.g., `"2017"`, `"2018"`).

Note: The arguments for `month` and `year` must be put between quotation marks.

Example

Today is May 16th, 2017. Let's say that I wanted to generate a report for this month so far. I set up the first `abbrvCounts.csv` file to go from September of 2016 up through April 2017, so this month is the first month a report can be generated. To generate this report, I would first set up my `channel` using the `odbcConnect()` code above, and then I would need to load in the `abbrvCounts.csv` data (remember to load it as `oldcounts`):

```
oldcounts <- read.csv("~/abbrvCounts.csv", stringsAsFactors=FALSE)
```

We can then look at the first few rows and columns of the data:

```
kable(oldcounts[c(1:6),c(1:6)])
```

word	n_spt16	n_oct16	n_nov16	n_dec16	n_jan17
pain	23408	23082	21937	21566	23129
pt	7964	8346	7860	7064	7664
left	5452	5154	4756	4469	4501
unspecified	4859	4684	4258	4408	4880
chest	4598	4601	4313	4569	4999
injury	4558	4623	4020	3609	3483

We can see counts for six words over five different months. For example, there were 4,999 uses of the word "chest" in admit reasons or chief complaints in January of 2017. It is very important to note at this point that *all counts generated only count ONE use of the word per patient visit at a facility.* If multiple messages were sent contaning the same text for a patient visit, all code utilized in this document counts each of the words *once*; similarly, if a message used a word twice in it (e.g., "pt complained of chest pain; pt felt sick"), the word "pt" would only be counted *once*.

Now that we (a) have set up a channel and (b) loaded in the data *with the name* `oldcounts`, then we are ready to generate a new report. The code is simple:

```
may <- newAbbrvReport(channel=channel, month="05", year="2017")
```

`may` returns a list of two data frames: one called `report` and another called `allcounts`. First, let's take a look at what a report looks like. This can be done by typing in `may$report`. This report does not fit on this page, so I will just take a look at one row (the 16th):

```
kable(may$report[16,])
```

	id	word	textraw
16	2017.05.11.3866_7933	cvf	Acute systolic CVF, new onset shortness of breath

The first column reports the BioSense ID, the second column the new abbreviation, and the third column the raw text where it came from. We can see that "CVF" had not been used in the previous 8 months, but has been used in the month of May. The full message read: "Acute systolic CVF, new onset shortness of breath." This report can be exported to a .csv, where it can be stored in the "reports" folder (located where this document is stored). To export, one executes the code:

```
write.csv(test$report, "report_may17.csv", row.names=FALSE)
```

`may` also contains a data frame called `allcounts`. This is an updated version of the `oldcounts` file you loaded at the beginning of the script. Let's compare the column names for `oldcounts` and `may$allcounts`:

```
names(oldcounts)

## [1] "word"    "n_spt16" "n_oct16" "n_nov16" "n_dec16" "n_jan17" "n_feb17"
## [8] "n_mar17" "n_apr17"

names(may$allcounts)

##  [1] "word"    "n_spt16" "n_oct16" "n_nov16" "n_dec16" "n_jan17" "n_feb17"
##  [8] "n_mar17" "n_apr17" "n_may17"
```

We can see that they are exactly the same, except that `may$allcounts` now includes a column for the number of times a word was used in May of 2017. This file can be downloaded as a .csv and replace the `abbrvCounts.csv` file that resides there now. Next month, this new file will become the file one uploads for `oldcounts` to compare for new words used in *June*. Then the report for June will generate an updated file, which can be saved as the new `abbrvCounts.csv` for July, etc. To save the new `abbrvCounts.csv` file, one executes the code:

```
write.csv(test$allcounts, "abbrvCounts.csv", row.names=FALSE)
```