prediction

Felipe Frazatto

10/5/2020

Loading Libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

Reading Database file

```
pF <- readRDS("./predFile.rds")
```

Define Side Functions

CleanText

Used to remove punctuation, numbers, elongated words and multiple blank spaces.

```
cleanText <- function(text) {

     newText <- text %>%
          tolower() %>%
          gsub(pattern = "[[:punct:]]", replacement =  " ") %>%
          gsub(pattern = "[0-9]", replacement = " ") %>%
          gsub(pattern = "\\b(?=\\w*(\\w){3}\\1)\\w+\\b", replacement = " ", perl = TRUE) %>%
          gsub(pattern = " +", replacement = " ")

     return(newText)

}
```

Prep

Prepares input string for the prediction. Cleans the string and separate it by word.

```
prep <- function(input.Str){

     # Initialize vector. Used for the string splititng into a word data frame
     words <- c()

     # Cleans String
     cleanStr <- cleanText(input.Str)

     # Counts number of words
     wordCount <- length(strsplit(cleanStr, " ")[[1]])

     # Populate words vector
     for(i in 1:wordCount){

          # Builds names following the format: "w" + i, where i = 1, 2, ..., n
          words <- c(words,
                     paste("w", i, sep = ""))

     }


     # Separates the cleaned input string into a word data frame
     strEval <- separate(data.frame(str = cleanStr),
                         str,
                         sep = " ",
                         into = words)

     return(strEval)

}
```

ngramFilter

Filters the training data frame with respect to a defined ngram and then to its words.

```
ngramFilter <- function(strEval, ngram){

     # Filters initial pF dataframe by the wanted ngram
     subSet <- filter(pF, n == ngram)

     # Calculates the input string size
     inputSize <- dim(strEval)[2]

     # Sequential filtering, word by word
     for(j in 1:(ngram - 1)){

          subSet <- subSet %>%
               filter(.[,j] == strEval[, inputSize - ngram + j + 1])

     }
```

```
    # Calculates the probability for each sequence to occur
    subSet$prob <- subSet$freq/sum(subSet$freq)

    return(subSet)

}
```

Prediction

pred

The predict function. Prepares the input, filters the data set to a more concise one, and get the probability of get a sequency.

Important to notice, if the ngramFilter function fail to find any thing like the input sequence it will return a 0 by 0 data frame, in this case the algorithm will sugest a period ".".

```
pred <- function(input.Str){

    # Prepares input string
    strEval <- prep(input.Str)

    # Load inital data set
    subSet <- pF

    # Finds the longest ngram in the data set
    ngramLim <- max(subSet$n)

    # Looks for a possible output prediction. Begins from the longest ngram
    # available to the shortest one (bigram). The loop will quit as soon as
    # a valid sequence is found.
    for(i in ngramLim:2){

        subSet <- ngramFilter(strEval, ngram = i)

        if(dim(subSet)[1] != 0)
        {
            break
        }

    }

    # If a sequence is found, suggest the word with highest probability (first
    # printed word) and picks randomly a second word, following the Mass
    # Probability Distribution calculated for the sequence.
    if(dim(subSet)[1] != 0){

        maxProb <- subSet[which.max(subSet$prob), max(subSet$n)]
        randProb <- sample(subSet[,max(subSet$n)], 1, prob = subSet$prob)

        output <- c(maxProb, randProb)

    }
```

```
    # If no valid sequence is found, suggests a period.
    else{output <- "."}

    return(output)

}
```

Exemples

```
pred("There")
```

```
## [1] "is" "s"
```

```
pred("There is")
```

```
## [1] "a" "a"
```

```
pred("There is a")
```

```
## [1] "good"  "great"
```

```
pred("There is a house")
```

```
## [1] "and" "in"
```

```
pred("There is a house in New")
```

```
## [1] "york" "york"
```

Conlusion

The algorithm is capable to predict some sentences, however it does not know the song The House of the Rising Sun by the Animals...