Text Mining - Module 12

Sooraj Shetty

April 17, 2017

Loading required packages

require(tm)

## Loading required package: tm

## Warning: package 'tm' was built under R version 3.3.3

## Loading required package: NLP

require(wordcloud)

## Loading required package: wordcloud

## Warning: package 'wordcloud' was built under R version 3.3.3

## Loading required package: RColorBrewer

require(RTextTools)

## Loading required package: RTextTools

## Warning: package 'RTextTools' was built under R version 3.3.3

## Loading required package: SparseM

##   
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':  
##   
## backsolve

require(SnowballC)

## Loading required package: SnowballC

##   
## Attaching package: 'SnowballC'

## The following objects are masked from 'package:RTextTools':  
##   
## getStemLanguages, wordStem

require(e1071)

## Loading required package: e1071

## Warning: package 'e1071' was built under R version 3.3.3

require(qdap)

## Loading required package: qdap

## Warning: package 'qdap' was built under R version 3.3.3

## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

## Warning: package 'qdapRegex' was built under R version 3.3.3

## Loading required package: qdapTools

## Warning: package 'qdapTools' was built under R version 3.3.3

# Data

For the purpose of this assignment I will be using Dr. Suess qoutes [from the lesson] to perform text mining.

quote <- c("You have brains in your head.",  
 "You have feet in your shoes.",   
 "You can steer yourself any direction you choose.",   
 "You're on your own.",   
 "And you know what you know.",   
 "And YOU are the one who'll decide where to go...")

Create a term by document matrix for the Dr. Suess quote. Assume each sentence is a new document.

for(i in seq(quote)){   
 quote[i] <- gsub("'re", " are", quote[i])   
 quote[i] <- gsub("'ll", " will", quote[i])  
}  
  
quote\_corpus <- Corpus(VectorSource(quote))  
# quote\_corpus2 <- Corpus(DataframeSource(data.frame(quote)))  
  
for(i in seq(quote\_corpus)){  
 writeLines(as.character(quote\_corpus[[i]]))  
}

## You have brains in your head.  
## You have feet in your shoes.  
## You can steer yourself any direction you choose.  
## You are on your own.  
## And you know what you know.  
## And YOU are the one who will decide where to go...

## Clean the corpus  
  
quotes\_clean <- tm\_map(quote\_corpus,removePunctuation) ## Remove the punctuations  
  
for(i in seq(quotes\_clean)){  
 writeLines(as.character(quotes\_clean[[i]]))  
}

## You have brains in your head  
## You have feet in your shoes  
## You can steer yourself any direction you choose  
## You are on your own  
## And you know what you know  
## And YOU are the one who will decide where to go

quotes\_clean <- tm\_map(quotes\_clean,content\_transformer(tolower)) ## Convert to lower case  
  
for(i in seq(quotes\_clean)){  
 writeLines(as.character(quotes\_clean[[i]]))  
}

## you have brains in your head  
## you have feet in your shoes  
## you can steer yourself any direction you choose  
## you are on your own  
## and you know what you know  
## and you are the one who will decide where to go

quotes\_clean <- tm\_map(quotes\_clean,removeWords,stopwords("english")) ## Remove the stop words  
  
for(i in seq(quotes\_clean)){  
 writeLines(as.character(quotes\_clean[[i]]))  
}

## brains head  
## feet shoes  
## can steer direction choose  
##   
## know know  
## one will decide go

quotes\_clean <- tm\_map(quotes\_clean,stripWhitespace) ## Remove whitespaces  
  
for(i in seq(quotes\_clean)){  
 writeLines(as.character(quotes\_clean[[i]]))  
}

## brains head  
## feet shoes  
## can steer direction choose  
##   
## know know  
## one will decide go

dtm <- DocumentTermMatrix(quotes\_clean) ## Creates the Document Term Matrix  
inspect(dtm)

## <<DocumentTermMatrix (documents: 6, terms: 12)>>  
## Non-/sparse entries: 12/60  
## Sparsity : 83%  
## Maximal term length: 9  
## Weighting : term frequency (tf)  
## Sample :  
## Terms  
## Docs brains can choose decide direction feet head know shoes steer  
## 1 1 0 0 0 0 0 1 0 0 0  
## 2 0 0 0 0 0 1 0 0 1 0  
## 3 0 1 1 0 1 0 0 0 0 1  
## 4 0 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 2 0 0  
## 6 0 0 0 1 0 0 0 0 0 0

# We can see empty documents in our Document Matrix, therefore we will reomve them and create the matrix again  
  
rowTotals <- apply(dtm , 1, sum)  
empty.rows <- as.vector(which(rowTotals==0))  
  
quotes\_clean <- quotes\_clean[-as.numeric(empty.rows)]  
  
dtm <- DocumentTermMatrix(quotes\_clean)  
inspect(dtm)

## <<DocumentTermMatrix (documents: 5, terms: 12)>>  
## Non-/sparse entries: 12/48  
## Sparsity : 80%  
## Maximal term length: 9  
## Weighting : term frequency (tf)  
## Sample :  
## Terms  
## Docs brains can choose decide direction feet head know shoes steer  
## 1 1 0 0 0 0 0 1 0 0 0  
## 2 0 0 0 0 0 1 0 0 1 0  
## 3 0 1 1 0 1 0 0 0 0 1  
## 4 0 0 0 0 0 0 0 2 0 0  
## 5 0 0 0 1 0 0 0 0 0 0

tdm <- TermDocumentMatrix(quotes\_clean) # Creates the Term Document Matrix  
inspect(tdm)

## <<TermDocumentMatrix (terms: 12, documents: 5)>>  
## Non-/sparse entries: 12/48  
## Sparsity : 80%  
## Maximal term length: 9  
## Weighting : term frequency (tf)  
## Sample :  
## Docs  
## Terms 1 2 3 4 5  
## brains 1 0 0 0 0  
## can 0 0 1 0 0  
## choose 0 0 1 0 0  
## decide 0 0 0 0 1  
## direction 0 0 1 0 0  
## feet 0 1 0 0 0  
## head 1 0 0 0 0  
## know 0 0 0 2 0  
## shoes 0 1 0 0 0  
## steer 0 0 1 0 0

* Calculate the tf-idf for the three terms in the text. Assume each sentence is a new document.

# Below is the term frequency (tf) for the documnet matrix  
  
inspect(dtm)

## <<DocumentTermMatrix (documents: 5, terms: 12)>>  
## Non-/sparse entries: 12/48  
## Sparsity : 80%  
## Maximal term length: 9  
## Weighting : term frequency (tf)  
## Sample :  
## Terms  
## Docs brains can choose decide direction feet head know shoes steer  
## 1 1 0 0 0 0 0 1 0 0 0  
## 2 0 0 0 0 0 1 0 0 1 0  
## 3 0 1 1 0 1 0 0 0 0 1  
## 4 0 0 0 0 0 0 0 2 0 0  
## 5 0 0 0 1 0 0 0 0 0 0

# Below is the normalized term frequency\*inverse document frequency (tf-idf)  
  
terms <- DocumentTermMatrix(quotes\_clean,control = list(weighting = function(x) weightTfIdf(x, normalize = TRUE)))  
inspect(terms)

## <<DocumentTermMatrix (documents: 5, terms: 12)>>  
## Non-/sparse entries: 12/48  
## Sparsity : 80%  
## Maximal term length: 9  
## Weighting : term frequency - inverse document frequency (normalized) (tf-idf)  
## Sample :  
## Terms  
## Docs brains can choose decide feet head know  
## 1 1.160964 0.000000 0.000000 0.000000 0.000000 1.160964 0.000000  
## 2 0.000000 0.000000 0.000000 0.000000 1.160964 0.000000 0.000000  
## 3 0.000000 0.580482 0.580482 0.000000 0.000000 0.000000 0.000000  
## 4 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 2.321928  
## 5 0.000000 0.000000 0.000000 0.773976 0.000000 0.000000 0.000000  
## Terms  
## Docs one shoes will  
## 1 0.000000 0.000000 0.000000  
## 2 0.000000 1.160964 0.000000  
## 3 0.000000 0.000000 0.000000  
## 4 0.000000 0.000000 0.000000  
## 5 0.773976 0.000000 0.773976

# Below is the non-normalized term frequency\*inverse document frequency (tf-idf). It just give the idf for all the terms in the document  
  
terms <- DocumentTermMatrix(quotes\_clean,control = list(weighting = function(x) weightTfIdf(x, normalize = FALSE)))  
inspect(terms)

## <<DocumentTermMatrix (documents: 5, terms: 12)>>  
## Non-/sparse entries: 12/48  
## Sparsity : 80%  
## Maximal term length: 9  
## Weighting : term frequency - inverse document frequency (tf-idf)  
## Sample :  
## Terms  
## Docs brains can choose decide direction feet head  
## 1 2.321928 0.000000 0.000000 0.000000 0.000000 0.000000 2.321928  
## 2 0.000000 0.000000 0.000000 0.000000 0.000000 2.321928 0.000000  
## 3 0.000000 2.321928 2.321928 0.000000 2.321928 0.000000 0.000000  
## 4 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
## 5 0.000000 0.000000 0.000000 2.321928 0.000000 0.000000 0.000000  
## Terms  
## Docs know shoes steer  
## 1 0.000000 0.000000 0.000000  
## 2 0.000000 2.321928 0.000000  
## 3 0.000000 0.000000 2.321928  
## 4 4.643856 0.000000 0.000000  
## 5 0.000000 0.000000 0.000000

* Write a regular expression to segment the quote into separate sentences.

quote <- c("You have brains in your head. You have feet in your shoes. You can steer yourself any direction you choose.  
 You're on your own. And you know what you know. And YOU are the one who'll decide where to go...")  
  
q1 <- unlist(strsplit(quote,"[.]"))  
  
for(i in seq(q1)){  
 q1 <- gsub("(^\\n)\\s+","",q1)   
 q1 <- gsub("^\\s+","",q1)  
}  
q1 <- q1[-c(grep("^$|^ $",q1))]  
  
print(q1)

## [1] "You have brains in your head"   
## [2] "You have feet in your shoes"   
## [3] "You can steer yourself any direction you choose"  
## [4] "You're on your own"   
## [5] "And you know what you know"   
## [6] "And YOU are the one who'll decide where to go"

* Write a regular expression to tokenize the quote.

q1 <- gsub("'re"," are",q1)  
q1 <- gsub("'ll"," will",q1)  
quote\_tokens <- unlist(strsplit(q1,"\\s"))  
quote\_tokens <- tolower(quote\_tokens)  
quote\_tokens <- unique(quote\_tokens)  
  
print(quote\_tokens)

## [1] "you" "have" "brains" "in" "your"   
## [6] "head" "feet" "shoes" "can" "steer"   
## [11] "yourself" "any" "direction" "choose" "are"   
## [16] "on" "own" "and" "know" "what"   
## [21] "the" "one" "who" "will" "decide"   
## [26] "where" "to" "go"

* Create a frequency signaure for the quote . Assume each sentence is a new document.

# Determining Frequency Signature  
  
mymatrix <- as.matrix(tdm)  
f1 <- sort(rowSums(mymatrix), decreasing=TRUE)  
  
# Below is the frequency of each term in the corpus  
f1

## know brains head feet shoes can choose   
## 2 1 1 1 1 1 1   
## direction steer decide one will   
## 1 1 1 1 1

The contingency table for a word pair (brains, know) will be calculated as follows.

Total number of words in the corpus = 13 Frequency of word 'brains' = 1 Frequency of word 'know' = 2 Co-occurence of the words 'brains' and 'know' = 0 Since these words don't occur together in any document of the corpus.

Contingency table for word pair (brains, know) will be represented as:

. brains !brains  
  
 . know 0 2  
  
 . !know 1 10

```

The contingency table for a word pair (brains, know) will be calculated as follows.

Total number of words in the corpus = 13 Frequency of word 'brains' = 1 Frequency of word 'know' = 2 Co-occurence of the words 'brains' and 'know' = 0 Since these words don't occur together in any document of the corpus.

Contingency table for word pair (brains, know) will be represented as:

. brains !brains  
  
 . know 0 2  
  
 . !know 1 10