WordCloud de todas as músicas de um artista

José Machado

24/02/2022

WordCloud de todas as músicas de um artista do site Vagalume® com textmining e webscraping

A algum tempo surgiu uma tread no twitter aonde estavam mostrando as palavras mais faladas por alguns artistas famosos. Com o aprendizado que estou realizando de R visualizei a possibilidade de automatizar a recuperação e visualização destes dados com as ferrametas de WebScapping e TextMining que tenho aprendido a usar. Este documento vai demonstrar meu processo de pensamento e execução para a ferramenta.

Carregando os pacotes necessários

```
library(rvest)
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(stringr)
library(tm)
```

```
## Carregando pacotes exigidos: NLP
```

```
library(SnowballC)
library(wordcloud)
```

```
## Carregando pacotes exigidos: RColorBrewer
```

```
library(RColorBrewer)
```

Fazendo a indicação da página do artista

Com o uso da função read_html conseguimos ler todo o código da página do artista. Por não ter muitos conhecimentos em html e css consegui apenas fazer a leitura no site do vagalume que tinha categorias mais bem separadas para cada página.

```
# Configuração do link do artista do site vagalume

link <- "https://www.vagalume.com.br/the-beatles/"

# Leitura do html da página do artista
page <- read_html(link)
```

Leitura e organização das músicas

Depois da leitura da página com todas as músicas do artista precisamos fazer a criação de um objeto contendo cada link de forma que possam ser passados para a leitura posterior. Com a função html_nodes e a utilização da extensão SelectorGadget no navegador conseguimos separar apenas o link de cada música e criar um vetor com todos os links.

```
# Leitura dos links das músicas

musicas_links <- page %>%
html_nodes("#alfabetMusicList .nameMusic") %>% # Pega o link da página de cada música
html_attr("href") %>%
paste("https://www.vagalume.com.br", ., sep = "") # Concatena o sufixo do link em cada referência recup

head(musicas_links)
```

```
## [1] "https://www.vagalume.com.br/the-beatles/a-day-in-the-life.html"
## [2] "https://www.vagalume.com.br/the-beatles/a-hard-days-night.html"
## [3] "https://www.vagalume.com.br/the-beatles/a-little-rhyme.html"
## [4] "https://www.vagalume.com.br/the-beatles/a-shot-of-rhythm-and-blues.html"
## [5] "https://www.vagalume.com.br/the-beatles/a-taste-of-honey.html"
## [6] "https://www.vagalume.com.br/the-beatles/across-the-universe.html"
```

Extração das letras de cada música

a função get_letras fará a leitura da página. Algo que tive problemas no começo foi com a função html_text que também faz a conversão para texto das tags html, porém ela ao retirar as tags de pulas a linha ela fazia com que as palavras finais e iniciais de cada linha ficassem juntas. Consegui resolver isso com a funçõa html_text2 que pega essas tags de pular linha. Dessa forma consigo após a leitura usar a função str_replace_all para adicionar espaços nesses campos.

Com a função criada usamos a função sapply para pegar todos os links conseguidos acima e aplicar a função get_letras, concatenando cada música em um vetor chamado “letras” que terá em cada posição a letra completa de uma música.

```r
# Função para extração das letras de cada música que os links foram conseguidos acima
get_letras <- function(musica_link){
  letras_page <- read_html(musica_link)
  todas_letras <- letras_page %>%
  html_node("#lyrics") %>%  # Indicação da tag aonde as letras estão na página
  html_text2(preserve_nbsp = TRUE) %>% # Conversão para texto das letras
  paste(collapse = " ")
}


letras <- sapply(musicas_links, FUN = get_letras, USE.NAMES = FALSE)
letras <- str_replace_all(letras, "\n", " ")
```

Limpeza e organização das palavras para o text mining

Com a criação de um dataframe com as letras realizamos uma série de funções para limpeza de pontuações e remoção de palavras irrelevantes para o estudo.

```r
#########    Conversão dos textos para wordcloud  ############

df <- as.data.frame(letras)

# Convertendo para corpus
dfcorpus <- Corpus(VectorSource(df))
class(dfcorpus)
```

```
## [1] "SimpleCorpus" "Corpus"
```

```r
# Convertendo para texto plano
dfcorpus <- tm_map(dfcorpus, PlainTextDocument)
```

```
## Warning in tm_map.SimpleCorpus(dfcorpus, PlainTextDocument): transformation
## drops documents
```

```r
# Removendo puntuação
dfcorpus <- tm_map(dfcorpus, removePunctuation)

# Remover stopwords Português
dfcorpus <- tm_map(dfcorpus, removeWords, stopwords("portuguese"))
dfcorpus <- tm_map(dfcorpus, removeWords, c("pra", "vai", "tá","ser","tudo"," que","não",
                                            "faz","vou","pro","ond","gent","todo","que ","que"))
dfcorpus <- tm_map(dfcorpus, removeWords, c("pra","this", "and", "dont", stopwords('english')))

# Convertendo palavras de diferentes versões em uma só
dfcorpus <- tm_map(dfcorpus, stemDocument)
```

Matriz de visualização para remover possíveis stopwords remanescentes

Agora com as palavras já organizadas criamos uma matriz para vizualização das palavras mais comuns e com isso visualizar se alguma palavra irrelevante ficou e adicionarmos nas stopwords acima.

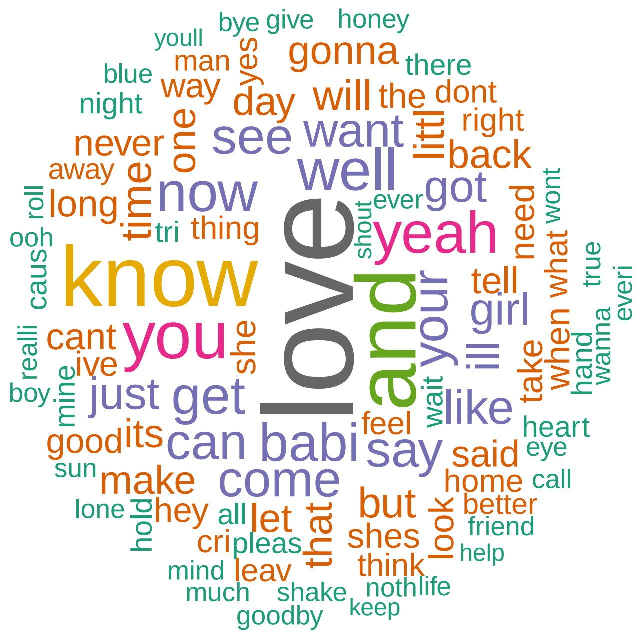
```r
# Cria uma matriz para vizualização das palavras
dtm <- TermDocumentMatrix(dfcorpus)
m <- as.matrix(dtm)
v <- sort(rowSums(m), decreasing = TRUE)
d <- data.frame(word = names(v), freq = v)
head(d, 20)
```

```
##      word freq
## love love  809
## know know  506
## and   and  499
## you   you  371
## yeah yeah  310
## well well  302
## now   now  286
## babi babi  275
## get   get  267
## come come  256
## your your  255
## can   can  254
## say   say  254
## see   see  242
## like like  231
## want want  224
## ill   ill  219
## got   got  215
## just just  215
## girl girl  212
```

Criação da wordCloud

Por fim é feita a wordcloud para uma visualização mais agradável de todas as palavras.

```r
# Criação do wordcloud
set.seed(1234)
wordcloud(dfcorpus,
 min.freq = 1, # Define a quantidade mínima de aparições da palavras para ela aparecer na wordcloud
 max.words = 100, # Define quantas palavras irão ser exibidas no máximo
 random.order = FALSE, # Define a ordem das palavras para decrescente
 rot.per = 0.2,  # Define a rotação das palavras
 colors = brewer.pal(8, "Dark2")) # Define a coloração
```


love
know
and
you
yeah
well
now
see
want
babi
can
get
just
come
say
like
girl
your
got
ill
gonna
day
will
the
dont
back
right
there
honey
give
bye
youll
man
yes
way
blue
night
never
one
littl
away
long
time
roll
ooh
caus
tri
thing
ever
shout
need
wont
what
true
everi
tell
when
take
hand
wanna
cant
ive
she
realli
boy
mine
good
its
feel
wait
heart
eye
said
make
sun
home
call
lone
hold
hey
all
let
that
but
look
better
friend
shes
cri
pleas
help
think
mind
leav
much
shake
noth
life
keep
goodby