Trabalho de Análise de Redes Sociais e Análise de Texto

Professor Fernando Ferreira

17/12/2019

Agenda

Buscaremos notícias relacionadas ao Banco BNDES.

Para tal acessaremos o portal de notícias G1.

As etapas dessa atividade serão:

- Acessaremos os links sobre as notícias do BNDES presentes na página de busca no portal;
- Acessaremos cada um dos links e recuperaremos as notícias;
- Analisaremos as notícias usando o pacote SpacyR;
- Criaremos uma lista de arestas para exportar para o Gephi;
- Criaremos uma visualização no Gephi e analisaremos as principais estatísticas para finalizar a tarefa.

Pacotes Necessários

Os pacotes que são importantes para a realização do trabalho:

```
set.seed(123)

#install the necessary packages
list.of.packages <- c('rvest',
                      'stringr',
                      'tidyverse',
                      'tm',
                      'igraph',
                      'wordcloud',
                      'urltools',
                      'spacyr',
                      'gtools')
new.packages <-
    list.of.packages[!(list.of.packages %in% installed.packages()[, "Package"])]
if (length(new.packages))
    install.packages(new.packages, dependencies = TRUE)


for (package in list.of.packages){
    library(package, character.only = TRUE)
}
```

Acessar as Páginas

Criaremos uma função auxiliar para acessar os links

```
scrape_post_links <- function(site) {
    # scrape HTML from input site
    source_html <- read_html(site)
    # grab the title attributes from link (anchor)
    # tags within H2 header tags
    links <- source_html %>%
        html_nodes("div.widget--info__text-container") %>%
        html_nodes("a") %>%
        html_attr("href")
    # filter out any titles that are NA (where no title was found)
    links <- links[!is.na(links)]
    # return vector of titles
    return(links)
}
```

Fazer iterações em 20 páginas

```
root <- "https://g1.globo.com/busca/?q=BNDES"

# get each webpage URL we need
all_pages <- c(root, paste0(root, "&page=", 1:20))
# use our function to scrape the title of each post
all_links <- lapply(all_pages, scrape_post_links)

# collapse the titles into a vector
all_links <- unlist(all_links)
```

A URL real está contida em parâmetro do link “u”

Criar uma função para extrair esses links

```
extract_urls <- function(raw_url) {
    params <- urltools::param_get(raw_url)
    scraped_url <- params$u
    return (url_decode(scraped_url))
}

cleaned_links <- lapply(all_links, extract_urls)
# Not interested in Videos from globoplay app
cleaned_links <- Filter(function(x) !any(grepl("globoplay", x)),
                        cleaned_links)
```

Acessar de cada link

```
scrape_post_body <- function(site) {
    # Escape 404 Not Found Errors
    try(
        text <- site %>%
            read_html %>%
            html_nodes("article") %>%
            html_nodes("p.content-text__container")  %>%
            html_text
    )
}

data <- lapply(cleaned_links, scrape_post_body)
data <- lapply(data,
               function(item) paste(unlist(item),
                                    collapse = ''))
```

```
# convert all titles to lowercase
cleaned <- tolower(data)
# remove any numbers from the titles
cleaned <- removeNumbers(cleaned)
# remove English stopwords
cleaned <- removeWords(cleaned, c(stopwords("pt"), "bndes", "banco"))
# remove punctuation
cleaned <- removePunctuation(cleaned)
 # remove spaces at the beginning and end of each title
cleaned <- str_trim(cleaned)
# convert vector of titles to a corpus
```

Pré-processar os textos para nuvens de palavras(1/2)

```
cleaned_corpus <- Corpus(VectorSource(cleaned))
# steam each word in each title
cleaned_corpus <- tm_map(cleaned_corpus, stemDocument)
doc_object <- TermDocumentMatrix(cleaned_corpus)
doc_matrix <- as.matrix(doc_object)
# get counts of each word
counts <- sort(rowSums(doc_matrix),decreasing=TRUE)
# filter out any words that contain non-letters
counts <- counts[grepl("^[a-z]+$", names(counts))]
# create data frame from word frequency info
frame_counts <- data.frame(word = names(counts), freq = counts)
```

Pré-processar os textos para nuvens de palavras (2/2)

Criar a nuvem de palavras

```
wordcloud(words = frame_counts$word,
          freq = frame_counts$freq,
          min.freq = 4,
          max.words=100, random.order=FALSE,
          rot.per=0,
          colors=brewer.pal(8, "Reds"))
```

Extrair todas as entidades dos textos

```
spacy_initialize(model="pt_core_news_sm")
entities <- spacy_extract_entity(unlist(data))
head(entities)
```

```
##   doc_id                                                          text
## 1  text1                                                A Renova Energia
## 2  text1 2ª Vara de Falências e Recuperações Judiciais de São Paulo
## 3  text1                                                            R$
## 4  text1                                                            R$
## 5  text1       Banco Nacional de Desenvolvimento Econômico e Social
## 6  text1                                                         BNDES
##   ent_type start_id length
## 1     MISC        2      3
## 2     MISC       19     10
## 3      LOC       44      1
## 4      PER       51      1
## 5      ORG       57      7
## 6      ORG       65      1
```

Criar lista de adjacência

Criaremos a lista de adjacência onde cada aresta define a coocorrência de duas entidades em um texto

```
# group entities by document
filtered_entities <- subset(entities, entities["ent_type"] == "ORG" |
                                      entities["ent_type"] == "PER" )
edges <- filtered_entities %>%
         group_by(doc_id) %>%
         summarise(entities = paste(text, collapse = ","))
# remove duplicated for the same document
edges <- lapply(str_split(edges$entities, ","),
                function(t){unique(unlist(t))})
# Auxiliary functions for creating adjancnt
get_adjacent_list <- function(edge_list) {
    adjacent_matrix <- combinations(length(edge_list),
                                    2, edge_list)
    return(adjacent_matrix)
}
adjacent_matrix <- edges %>%
                   lapply(get_adjacent_list) %>%
                   reduce(rbind)
```

Criar Objeto Grafo que será exportado por Gephi

```
df <- as_tibble(adjacent_matrix, colnames=c('source', 'target'))
weighted_edgelist <- df %>%
                     group_by(V1, V2) %>%
                     summarise(weight=n())
news_graph <- weighted_edgelist %>%  graph_from_data_frame(directed=F)
write_graph(news_graph, 'news_graph.graphml', 'graphml')
```