tweet-urlscrapper

This R code is useful for examing news shared by twitter users. The functions provided:

- Expand shortened urls embedded in tweets
- Provide the news source
- Scrap the web for text from the news article

You may apply functions to vector objects or a data frame of tweets. Code was created consider the packages `rtweet` and `streamR`. However, the functions work on any data frame. If using a dataframe, tweets must be stored in a column entitled *text*.

`getURL` creates a data frame containing all of the urls shared the corresponding tweet. Since some tweets contain multiple urls, the data frame may contain duplicate tweets (but it will not duplicate matches of the tweet and the url).

Example for url and source detection

Tweets were collected using `rtweet`. Sample of 87 tweets containing the keywords *Trump* or *Obama*.

```
tweets <- read.csv("https://raw.githubusercontent.com/marisaasmith/tweet-urlscrapper/master/tweets-examp
```

Run `fullUrl`, `getSource`, and `getUrl` functions

```
# faster code for getSource()
getSource <- function(df){
  df %>%
    mutate(source=full_url %>%
             stringr::str_replace("^(https?:\\/\\/)?(www\\.)?", "") %>%
             stringr::str_extract("([\\da-z\\.-]+)\\.([a-z\\.]{2,6})") %>%
             stringr::str_replace("^[am]{1}\\.", ""))
}
```

```
getUrl <- function(df){
  require(tidyverse)
  df %>%
    mutate(url = stringr::str_extract_all(text, "https://t.co/[a-z,A-Z,0-9]*")) %>%
    tidyr::unnest(url) %>% rowwise() %>%
    mutate(full_url = ifelse(stringr::str_detect(url,"https://t.co/[a-z,A-Z,0-9]*", negate = F) == T,
                             {httr::GET(url) %>% magrittr::use_series("url")},
                             NA)) %>% group_by(full_url) %>%
    distinct() %>% select(-url)
}
```

`getUrl` adds column of expanded urls (*full_url*) and the source (*source*)

```
tweets <- getUrl(tweets)
```

Example for web scrapping

`getText` uses `rvest` to shared scrap news articles. **NOTE:** The code uses generic xPath 'p' for webscrapping. Depending on the website, you may want to change the path. I recommend the Google Chrome selector gadget

```
getText <- function(url_vector, source_vector){
  require(rvest)
  require(tidyverse)
  source_vector <- test$source %>%
  {grep("twitter", ., value = TRUE, invert = T)}%>%
```

```
  { grep("youtube", ., value = TRUE, invert = T) }
  url_vector <- url_vector %>% {grep("twitter", ., value = TRUE, invert = T)} %>%
  {grep("youtube", ., value = TRUE, invert = T)}
  url_count <-  1:length(url_vector)
  article.text.full <- {}
  for (i in url_count) {
    article.text  <- data_frame(text = read_html(url_vector[i]) %>%
                                  html_nodes("p") %>%
                                  html_text())

    Article.withID <- cbind(article.text, source_vector[i])
    Article.withID <- as.tibble(Article.withID)


    article.text.full <- bind_rows(article.text.full, Article.withID)
  }
  return(article.text.full)

}
```

`getText` returns a data frame of text and the corresponding source (excluding twitter and youtube sources)

```
getText(tweets$full.url, tweets$source)
```

Eventually, I plan to create a package with these and similar functions that assists social media news analysis.