Script Data Management

Gabriele Carrara and Alberto Filosa

06/07/2020

Contents

1 Libraries

R libraries:

```
library(rvest)
library(lubridate)
library(readr)
library(dplyr)
library(reticulate)
library(stringr)
library(ggplot2)
library(jsonlite)
```

Python libraries:

```
#-- Imported os to scrape old tweet with GetOldTweets3 package
import os
os.system("pip install GetOldTweets3")
import GetOldTweets3
import pandas as pd
import datetime as dt
import sys
import meaningcloud
from kafka import KafkaProducer
from kafka import KafkaConsumer
import json
import time
import random
from pymongo import MongoClient
```

Created environment Conda to use python in R:

```
reticulate::use_condaenv("r-reticulate", required = TRUE)

#### Install Python libraries:
#reticulate::py_install("pandas")

#### Info at:
#### https://abndistro.com/post/2018/01/09/getting-started-with-python-in-r-markdown-using
#### -the-reticulate-package/
```

2 Scraping

2.1 Weather

Scraped weather data from ilmeteo.it for Milano, Palermo, Napoli and Roma in 2019:

```
for (provincia in province)
{
  i = 1
  for (mese in mesi)
  {
    #-- Change provincia and mese in URL website
    url <- paste0('https://www.ilmeteo.it/portale/archivio-meteo/', provincia, '/2019/', mese)
    webpage <- read_html(url)

    #-- Scrape weather data from xpath and transform it in a table
    temp <- webpage %>%
            html_nodes(xpath = '//*[@id="mainc"]/div[1]/table[2]') %>%
            html_table()

    #-- Condition because there aren't any weather data in August, September and October
    #-- In Milan
    if((provincia != "Milano") || (!(mese %in% c("Agosto", "Settembre", "Ottobre"))))
    {
      temp <- temp[[1]]
```

```
      temp["Provincia"] = provincia
      temp["Mese"]      = mese
      temp["NumMese"]   = i
      meteo <- bind_rows(meteo, temp)
    }

    i = i+1

  }
}
head(meteo)
```

Extraction of important columns:

```
meteo <- meteo[, c("Giorno", "T Media", "T min", "T max", "Provincia", "Mese", "NumMese")]
meteo["Anno"] = 2019
```

Created lists to get the tweets:

```
#-- Get list of words, cities and [lat, lon] to get the tweets
lista = ["freddo",
         "sole",
         "nuvole",
         "pioggia",
         "neve",
         "temperatura",
         "caldo"]

citta = ["'45.28, 09.10'",
         "'41.54, 12.28'",
         "'40.51, 14.14'",
         "'38.06, 13.20'"]

nomi  = ["Milano",
         "Roma",
         "Napoli",
         "Palermo"]

df    = pd.DataFrame()
```

2.2 Tweets

Scraping tweets:

```
for i in range(0,4):
    for el in lista:
        cit = citta[i]

        #-- Query to search old tweets
        #-- $el is used to pass element in list and $cit to pass element of the cities
        os.system("GetOldTweets3 --querysearch $el --since 2019-01-01 --until 2019-12-31 \
                  --lang it --near $cit --within 100km")
```

```
        #-- Read with pandas the outputs
        df_temp = pd.read_csv("/home/jovyan/Progetto-DM/output_got.csv")

        #-- Add city and words used to get tweets
        df_temp["citta"]  = nomi[i]
        df_temp["parola"] = el

        #-- Concatenate tweets
        df = pd.concat([df, df_temp])
```

```
#-- Written json file to import in mongo
df.to_json("/home/jovyan/Progetto-DM/tweets.json", orient = "records")

#-- Client object to access the database
client = MongoClient("mongo", 27017, username = "admin", password = "DataMan2019!")

#-- Query for importing json file in MongoDB
!mongoimport --authenticationDatabase=admin --authenticationMechanism=SCRAM-SHA-1
--username=admin --password=DataMan2019! --db=DataMan --collection=tweets
--file=/data/my-data/tweets.json --jsonArray
```

Downloaded csv to integrate missing data:

```
#-- We downloaded this csv from https://rp5.ru/Weather_in_the_world website
#-- With the missing data
df810 = pd.read_csv("/home/alberto/Scaricati/s.csv", skiprows = 6, sep = ",")
df810.head()

#-- We keep only local time and temperature
df810 = df810[["Local time in Milan (airport)","T"]]
df810.columns = ["Data", "T"]
df810.head()

#-- We split the data keeping only date (not hours)
df810['Data'] = df810['Data'].str.split(" ").str[0]
df810.head()
```

Grouping last dataset to one daily temperature:

```
#-- Group by and calculate mean, max and min temperature every day
Tmean = df810.groupby("Data").mean()
Tmean = Tmean.reset_index()
Tmean.columns = ["Data", "T Media"]
Tmean.shape

Tmin = df810.groupby("Data").min()
Tmin = Tmin.reset_index()
Tmin.columns = ["Data", "T min"]

Tmax = df810.groupby("Data").max()
Tmax = Tmax.reset_index()
Tmax.columns = ["Data", "T max"]
```

```
#-- Merged the three temperatures in one dataset
df = pd.merge(Tmean, Tmin, on = "Data")
df = pd.merge(df, Tmax, on = "Data")
df.head()
```

Transformation of dates in YYYY-MM-DD format:

```
#-- Rounded and transformed temperature as integer
df = round(df, 0)
df[["T Media", "T min", "T max"]] = df[["T Media", "T min", "T max"]].astype("int")

#-- Added province of Milan
df["Provincia"] = "Milano"

#-- Transformed date in YYYY-MM-DD for all the datasets
df["Data"] = pd.to_datetime(df["Data"], format = "%d.%m.%Y")
df["Data"] = df["Data"].apply(lambda x: dt.datetime.strftime(x, "%Y-%m-%d"))

df.tail()
```

Mongo Import:

```
#-- Written json file to import in mongo
df.to_json("/home/jovyan/Progetto-DM/namilano.json", orient = "records")

#-- Query for importing json file in MongoDB
!mongoimport --authenticationDatabase=admin --authenticationMechanism=SCRAM-SHA-1
--username=admin --password=DataMan2019! --db=DataMan --collection=namilano
--file=/data/my-data/namilano.json --jsonArray
```

Concatenation of datasets:

```
#-- Created date from three different columns
meteo$Data <- as.Date(with(meteo, paste(Giorno, NumMese, Anno, sep = ".")), "%d.%m.%Y")
meteo <- meteo[, c("Data", "T Media", "T min", "T max", "Provincia")]
meteo

#-- Used python dataset and imported in R
df <- py$df
df$Data <- as.Date(df$Data)

#-- Changed variable types in integer
meteo$`T Media` <- as.integer(as.character(str_replace(meteo$`T Media`, ' °C','')))
meteo$`T min` <- as.integer(as.character(str_replace(meteo$`T min`, ' °C','')))
meteo$`T max` <- as.integer(as.character(str_replace(meteo$`T max`, ' °C','')))

#-- Added data for period 11/10/2019-26/10/2019 in Milan which missed
#-- In both previous datasets
ott <- read.csv("/home/alberto/Scaricati/OttobreNA.csv", header = T, sep = ",")
colnames(ott) <- c("Data", "T_Media", "T_Min", "T_Max", "Provincia")

ott$Data      <- as.Date(ott$Data)
```

```
ott$Provincia <- as.character(ott$Provincia)
ott$T_Media   <- as.numeric(ott$T_Media)
ott$T_Min     <- as.numeric(ott$T_Min)
ott$T_Max     <- as.numeric(ott$T_Max)

#-- Concatenated three datasets (meteo, df and ott)
meteo_def <- bind_rows(meteo,df)
colnames(meteo_def) <- c("Data", "T_Media", "T_Min", "T_Max", "Provincia")
meteo_def <- bind_rows(meteo_def, ott)
meteo_def <- meteo_def[order(meteo_def[, 5], meteo_def[, 1]), ]

#-- Managed 2019-02-05 missing data, imputateed with previous data
which(is.na(meteo_def[,2]))

meteo_def[36, c("T_Media", "T_Min", "T_Max")] <- c(5, 1, 11)
meteo_def[400, ]

meteo_def[401, c("T_Media", "T_Min", "T_Max")] <- c(10, 6, 13)
meteo_def[765, ]

meteo_def[766, c("T_Media", "T_Min", "T_Max")] <- c(10, 7, 12)
meteo_def[1130, ]

meteo_def[1131 ,c("T_Media", "T_Min", "T_Max")] <- c(9, 4, 15)
which(is.na(meteo_def[, 2]))
```

Mongo Import:

```
#-- Written json file to import in mongo
meteo_def.to_json("/home/jovyan/Progetto-DM/meteo_def.json", orient = "records")

#-- Query for importing json file in MongoDB
!mongoimport --authenticationDatabase=admin --authenticationMechanism=SCRAM-SHA-1
--username=admin --password=DataMan2019! --db=DataMan --collection=meteo_def
--file=/data/my-data/meteo_def.json --jsonArray
```

Final dataset with complete values: each row is a tweet with date and temperature of the date

```
#-- DataMan database in mongo
db = client.DataMan

#-- Meteo_def collection in DataMan database
meteo_def = db.meteo_def

#-- Tweets collection in DataMan database
tweets = db.tweets
```

```
#-- Read only weather and tweets datasets
meteo_def <- py$meteo_def[, 2:6]
meteo_def$Data <- as.Date(meteo_def$Data)

tweets <- py$tweets
```

```
#-- Filter from not interesting user tweets
tweets <- filter(tweets, username != "MeteoPregnana")

#-- Keep only interesting variables
tweets <- tweets[, c("date","text","citta","parola")]
colnames(tweets) <- c("Data","Testo","Provincia","Parola")

#-- Changed variable types
tweets$Data  <- as.Date(tweets$Data)
tweets$Testo <- as.character(tweets$Testo)

#-- Merged datasets
df_def <- merge(meteo_def, tweets,by = c("Data", "Provincia"))
```

Mongo Import:

```
#-- Written json file to import in mongo
df.to_json("/home/jovyan/Progetto-DM/tweets_weather.json", orient = "records")

#-- Query for importing json file in MongoDB
!mongoimport --authenticationDatabase=admin --authenticationMechanism=SCRAM-SHA-1
--username=admin --password=DataMan2019! --db=DataMan --collection=tweets_weather
--file=/data/my-data/tweets_weather.json --jsonArray
```

3 Exploratory Analysis

```
load("~/Documenti/Progetto-DM/Meteo_Tweet.Rdata")
```

Missing Values:

```
sapply(df_def,function(x)(sum(is.na(x))))

# no missing values
which(is.na(df_def[,2]))
```

Boxplot:

```
#-- Statistical summaries of weathers data
summary(meteo_def)

#-- Boxplot of temperatures
boxplot(meteo_def[,c(3,2,4)])
```

Checks for temperatures:

```
#-- Check if minimal temperatures less than mean temperatures
x <- (meteo_def$T_Min <= meteo_def$T_Media)
table(x)
```

```r
#-- Check if maximum temperatures over than mean temperatures
x <- (meteo_def$T_Max >= meteo_def$T_Media)
table(x)
```

Line plot:

```r
#-- Line plot of mean (black), min (blue), max (red ) temperatures for each province
ggplot(meteo_def, aes(x=Data)) +
  facet_wrap(meteo_def$Provincia) +
  geom_line(data=meteo_def, aes(y=T_Min), colour="blue") +
  geom_line(data=meteo_def, aes(y=T_Media), colour="black") +
  geom_line(data=meteo_def ,aes(y=T_Max), colour="red") +
  ylab(" ")
#-- Data seems coherent
```

Off topic tweets:

```r
#-- Get a sample of 10
samp <- sample(nrow(df_def), 100)

#-- Data in the sample
sample <- as.data.frame(df_def[samp, "Testo"])
#-- 32% of 100 tweet off topic
```

4 Sentiment Analysis

Read and removed comments, tags, ect. in tweets:

```python
#-- API keys MeaningCloud for Sentiment
license_key = "d207e296ee26998b8710db5d933e1f0f"

#-- Read datasets
df_def = db.tweets_weather

#-- Remove tags
df_def.Testo = df_def.Testo.replace("@[\w]*[_-]*[\w]*"," ",regex=True)

#-- Remove url links
df_def.Testo = df_def.Testo.replace("https?://[\w/%-.]*"," ",regex=True)

#-- Remove extra spaces
df_def.Testo = df_def.Testo.replace('\s+', ' ',regex=True)

#-- Remove initial spaces
df_def.Testo = df_def.Testo.replace('^ ', '', regex=True)

#-- Remove final spaces
df_def.Testo = df_def.Testo.replace(' $', '', regex=True)

#-- All lower
df_def.Testo = df_def.Testo.apply(lambda x: x.lower())
```

```
#-- Tokenized
df_def.Testo = df_def.Testo.apply(lambda x: x.strip())

# reset index for column tweets
text = df_def["Testo"][0:27000].reset_index()["Testo"]
```

Sentiment analysis:

```
response = [] #-- Empty list for sentiment

for i in range(0,len(text)):
    testo = text[i]
    #-- API sentiment analysis
    sentiment_response = meaningcloud.SentimentResponse(
                         meaningcloud.SentimentRequest(license_key, lang = 'it',
                         txt = testo, txtf = "plain").sendReq())

    if sentiment_response.isSuccessful():
        #-- Add sentiment response to list
        response.append(sentiment_response.getGlobalScoreTag())

    else:
        #-- Add None in case of error
        response.append("None")
```

Mongo Import:

```
#-- Written json file to import in mongo
df_def.to_json("/home/jovyan/Progetto-DM/DfSentiment.json", orient = "records")

#-- Query for importing json file in MongoDB
!mongoimport --authenticationDatabase=admin --authenticationMechanism=SCRAM-SHA-1
--username=admin --password=DataMan2019! --db=DataMan --collection=tweets_weather_sentiment
--file=/data/my-data/DfSentiment.json --jsonArray
```

5 Velocity

5.1 Producer

Kafka Producer:

```
#-- Created a Kafka Producer
producer = KafkaProducer(bootstrap_servers = ["kafka:9092"],
                         value_serializer  = lambda v: json.dumps(v).encode("utf-8"))
```

Sent message:

```
for i in range(0, 266):
    #-- Token 100 tweets and transformed to JSON
    df_def = db.tweets_weather_sentiment
```

```
    tweet_json = df_def.loc[i*100:(i+1)*99,:].to_json(orient = "records",
                                                      lines  = True, force_ascii = False)
    #-- Sent to broker
    producer.send(topic = "tweet_meteo", value = tweet_json)

    #-- Pseudo-random waiting
    time.sleep(random.randint(10,20))
```

5.2 Consumer

Kafka Consumer:

```
#-- Created a Kafka Consumer
consumer = KafkaConsumer(bootstrap_servers  = ["kafka:9092"],
                         auto_offset_reset  = "latest",
                         value_deserializer = lambda v: json.loads(v.decode("utf-8")))

#-- To our topic
consumer.subscribe(["tweet_meteo"])
```

Received message and concatenated in tweets dataframe:

```
#-- Created an empty dataframe
tweets = pd.DataFrame()

for message in consumer:

    #-- Read message from Kafka and extract value
    message = message.value

    #-- Added to dataframe
    temp   = pd.read_json(message, orient = "records", lines = True, encoding = "utf-8")
    tweets = pd.concat([tweets, temp])
```

6 MongoDB

JSON file of `df_def`:

```
#-- Written json file to import in mongo
df_def.to_json("/home/jovyan/Progetto-DM/DfSentiment.json", orient = "records")

#-- Query for importing json file in MongoDB
!mongoimport --authenticationDatabase=admin --authenticationMechanism=SCRAM-SHA-1
--username=admin --password=DataMan2019! --db=DataMan --collection=weather_tweets_sentiment_final
--file=/data/my-data/DfSentiment.json --jsonArray
```