TEXT MINING sobre datos recolectados de Twitter.

ZEVALLOS, JOSE[1,*], FAZIO, BORIS[1,], AND HUERTAS, ANTHONY[1,***]**

[1] *Escuela de Posgrado, Pontificia Universidad Católica del Perú, Lima, Perú*
[*] *Cod: 20173861*
[**] *Cod 20173896*
[**] *Cod: 20173728*

Compiled May 17, 2018

Mineria de datos

1. INTRODUCCIÓN

Text Mining es un mecanismo por el cual se transforman datos cualitativos e inestructurados en datos de bueno uso por una computadora. Estos datos inestructurados no pueden ser medidos numéricamente; sin embargo, presentan características de color, textura y descripción textual, esta última será en la cual se verá enfocado el presente proyecto. En conclusión, text mining nos permite generar nueva información, provenientes de una amplia colección de datos, con el objetivo puedan ser examinados.Exiten diversos análisis que se pueden realizar en Text mining; sin embargo, en este proyecto se harán uso del **análisis de sentimientos** y nube de palabras, también conocido como **Workcloud**.

El análisis Workcloud es una representación gráfica de palabras frecuentemente usadas en una colección de datos. El tamaño de cada palabra en el gráfico ejecutado por Workcloud es un indicador de la frecuencia de dicha palabra en el total de nuestros datos. Esto quiere decir, que mientras mayor sea la frecuencia en la que se presenta una palabra, se visualizará de mayor tamaño.

El análisis de sentimientos es una medida rescatada de los términos positivos y negativos encontrados en una descripción textual, y mediante ella permitir determinar si en general las expresiones son negativas, positivas o neutrales respecto a un tema en particular. Este análisis toma una relevancia con el criterio de comparación pues permite contrastar el entorno de sentimientos alrededor de una situación presente.

En el presente trabajo se recolectó información de la app **Twitter** utilizando el interfaz de programación provisto por la empresa, diseñando un bot mediante el lenguaje de programación Python con inclusión de la librería tweepy necesaria para trabajar con este tipo de datos. Esta librería de Python provee de las funcionalidades RESTfull de Twitter. Los paquetes adicionales de uso necesario en el diseño del bot, y análisis de ser conveniente, serán las siguientes:

Listing 1. Librerías de Python.

```
import sys,tweepy,re
import matplotlib.pyplot as plt
import dataset
import json
from textblob import TextBlob
from sqlalchemy.exc import ProgrammingError
from tweepy.streaming import StreamListener
```

El diseño de programación del bot posee dos condiciones: tener una cuenta de Twitter y una aplicación registrada en https://apps.twitter.com. Al hacerlo se te provee de 4 claves de acceso (*Consumer Key, Consumer Secret, Access Token y Access Level*), las cuales tendrán que ser almacenadas en un script alterno con el objetivo se mantenga la privacidad del investigador.

Listing 2. Función API con autentificacion.

```
# Codigos de acceso de la App de Twitter del Usuario

# Consume:
CONSUMER_KEY    = 'XXXXXXXX'
CONSUMER_SECRET = 'XXXXXXXX'

# Access:
ACCESS_TOKEN  = 'XXXXXXXX'
ACCESS_SECRET = 'XXXXXXXX'

# ...
CONNECTION_STRING = "sqlite:///tweets.db"
TABLE_NAME = "tabla_nombre"
```

2. ANÁLISIS DE DATOS CON FRASES REFERENTES AL PARTIDO “BAYERN MUNICH VS REAL MADRID” DEL 25/04/18.

Se recolectaron tweets, los cuales fueron almacenados en una base de datos SQLite. Adicionalmente, por cada tweet se guardó su idioma, el id del usuario que lo realizó, y cuantos retweets se dieron. El bote ha sido diseñado mediante el lenguaje de programación Python, y el procesamiento mediante el lenguaje de programación R.

A. Diseño del bot de recolección de datos

Se programa el bot con un mecanismo de recolección de tweets, mediante cierta especificación de palabras clave (keywords), en tiempo real 5 minutos previos, durante y 5 minutos después del partido por semifinales de la Champions League 2018 entre los equipos Bayern Munich y Real Madrid.

Para ello se diseña una primera clase de clasificacion y procesamiento de tweets en tiempo real

Listing 3. Bot: Clase TwitterStreamer().

```
from credentials import *

db = dataset.connect(CONNECTION_STRING)

class TwitterStreamer():
        def __init__(self):
        pass

        def stream_tweets(self, fetched_tweets_filename, hash_tag_list):

                # API con autentificacion y codigos de acceso
                listener = StdOutListener(fetched_tweets_filename)
                auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
                auth.set_access_token(ACCESS_TOKEN,ACCESS_TOKEN_SECRET)
                stream = Stream(auth, listener)

                # Filtro de captura de datos por palabras clave
                stream.filter(track=hash_tag_list)
```

Se diseña una segunda clase, la cual permite ejecutar los tweets que se desean evaluar y ciertas especificaciones como las coordenadas, el texto, el ID, el numero de retweets y el lenguaje.

Listing 4. Bot: Clase StdOutListener().

```
class StdOutListener(StreamListener):
        def __init__(self, fetched_tweets_filename):
                self.fetched_tweets_filename = fetched_tweets_filename

        def on_data(self, data):

                table = db[twitter_credentials.TABLE_NAME]
                try:
                        #print data
                        all_data = json.loads(data)
                        text = all_data['text']
                        ID = all_data["id_str"]
                        coords=all_data['coordinates']
                        retweets=all_data["retweet_count"]
                        lenguaje=all_data['lang']

                        table.insert(dict(
                                coordinates=coords,
                                text=text,
                                id_str=ID,
                                retweet_count=retweets,
                                lengua=lenguaje,
                        ))

                        with open(self.fetched_tweets_filename, 'a') as tf:
                                tf.write(data)
                        return True
                except BaseException as e:
                        print("Error on_data %s" % str(e))
                return True

        def on_error(self, status_code):
                if status_code == 420:
                        #returning False in on_data disconnects the stream
                        return False

        def on_status(self, status):
                all_data = json.loads(status)
                tweet = all_data['text']
```

En esta última parte del diseño se introducen los keywords que deberán incluir los tweets, los cuales son : “Bayern Munich”, “Real Madrid”,“Cristiano Ronaldo” y “Robben”. Posteriormente el nombre del archivo de almacenamiento *tweets5.txt* que por la clase StdOutListener será obtenido en formato *.db*.

Listing 5. Bot: Keywords y almacenamiento.

```
if __name__ == '__main__':

        # Authenticate using config.py and connect to Twitter Streaming API.
        hash_tag_list = ['Bayern Munich','Real Madrid','Cristiano Ronaldo','Robben']
        fetched_tweets_filename = "tweets5.txt"

        twitter_streamer = TwitterStreamer()
        twitter_streamer.stream_tweets(fetched_tweets_filename, hash_tag_list)
```

B. Convertir base de datos en formato .db (SQL) a .csv

En este proceso se desarrolla un código en R que permite convertir la base de datos en formato SQL obtenida por el bot, a un formato .csv con el objetivo puede facilitarse el análisis mediante funciones más generales. Además se adiciona un proceso de limpieza de datos en la cual se retiran emojis y tildes de las letras. Los datos son almacenados en un archivo llamado *tweets.csv*

Listing 6. Conversión de data SQL a *.csv*.

```
library(RSQLite)

# Cargar BD SQL
con <- dbConnect(SQLite(), dbname="tweets5.db")
# Ver tablas en la BD
sq <- dbSendQuery(con, "SELECT name FROM sqlite_master WHERE type='table'")
tn <- dbFetch(sq, n = -1)
# Ver filas en la tabla
sq <- dbSendQuery(con, paste0("SELECT * FROM ",tn[1,]))
d <- dbFetch(sq, n = -1)
dbDisconnect(con)

# Limpieza + Exportacion
d$text %<>%
        iconv("UTF-8", "ASCII//TRANSLIT") %>%
        ifelse(is.na(.), iconv(d$text, "latin1", "UTF-8"), .) %>%
        stri_replace_all(regex="[^[:graph:]]", replacement=" ") %>%
        stri_replace_all(regex="(f|ht)tp\\S+\\s*", replacement=" ") %>%
        stri_replace_all(regex="[,?.:|]", replacement=" ") %>%
        textclean::replace_emoji(emoji_dt = data.table(lexicon::hash_emojis[,1], y=" "))
write.csv(d, "tweets.csv", row.names = FALSE)
```

C. Preprocesamiento de la data

Se diseña un mecanismo de preprocesamiento de los datos previo al análisis de conglomerados y reglas de asosiación que serán de interés posterior. Los resultados de este proceso desarrollado en R, los cuales serán almacenados en *02_output.Rdata*, son los siguientes:

- **resumen**: Número de tweets en total (tt), tweets que son un retweet (rt), tweets con una llamada a otro @usuario (at), tweets con al menos un hashtag (ht), y tweets con llamada @ y hashtag (ht.at), para cada idioma con >2k tweets totales.
- **rt.df**: Tabla con el número de veces que se repite cada retweet para un idioma dado, servirá para mostrar los tweets mas populares.
- **at.df, ht.df**: Tabla que muestra @usuarios y #hashtags mas populares, sin contar los retweets.
- **d.stop**: Cinco corpus (uno por idioma) sin stopwords, con la presencia de cada palabra en un tweet representada en formato matriz. Servirá para el análisis de conglomerados (tweets similares) o reglas de asociacion (frases comunes).
- **d.stem**: Mecanismo similar a **d.stop** pero con reducción de las palabras a su raiz.

Listing 7. Preprocesamiento.

```
library(magrittr)
library(stringi)
library(textclean)
library(data.table)
library(SnowballC)
library(tm)
options(stringsAsFactors = FALSE)

# Cargar datos + Preprocesamiento general (Solo lenguas > 2k tweets)
```

```r
d <- read.csv("tweets.csv", stringsAsFactors = FALSE)[,c(1:3,5)]
d <- d[d$lengua %>% is.element(c("en","es","pt","fr","de")),]

# Retirar blanco en los extremos de cada tweet + regex ( @ + # + RT .*)
d$text %<>% trimws
re.at <- "@[[:alnum:]_]+"
re.ht <- "#[[:alpha:]][[:alnum:]_]+"
re.rt <- "RT .*"

# Resumenes
# Taggeo de tweets [RT|@|#]
# RT
d$text %>% stri_match_first(regex=re.rt) %>% is.na %>% not -> d$rt
# @
d$text[!d$rt] %>% stri_match_first(regex=re.at) %>% is.na %>% not -> d$at[!d$rt]
# #
d$text[!d$rt] %>% stri_match_first(regex=re.ht) %>% is.na %>% not -> d$ht[!d$rt]

# Conteo de RTs
d$text[d$rt] %>% table -> rt.tab
rt.df <- data.frame(text=names(rt.tab),n.rt=as.vector(rt.tab))
d.rt <- merge(d, rt.df, by = "text")[,c("text","lengua","n.rt")]
rt.df <- d.rt[!duplicated(d.rt),]
rm(d.rt)

# Hashtags por idioma
d$lengua %>% unique %>% lapply(
        function(x){
                d[d$lengua==x,"text"] %>% stri_extract_all(regex=re.ht) %>% unlist %>% table %>%
                        sort(decreasing = TRUE) -> ht.tab
                data.frame(ht=names(ht.tab), n=as.vector(ht.tab), lengua=x)
        }) %>% do.call(rbind,.) -> ht.df

# Usuarios por idioma
d$lengua %>% unique %>% lapply(
        function(x){
                d[d$lengua==x,"text"] %>% stri_extract_all(regex=re.at) %>% unlist %>% table %>%
                        sort(decreasing = TRUE) -> at.tab
                data.frame(at=names(at.tab), n=as.vector(at.tab), lengua=x)
        }) %>% do.call(rbind,.) -> at.df

# Limpiar palabras
d.tm <- d[!d$rt,c("lengua","text")]
d.tm$text %<>%
        stri_replace_all(regex=re.at, replacement = " ") %>%
        stri_replace_all(regex=re.ht, replacement = " ") %>%
        stri_replace_all(regex="<[:alnum:][:alnum:]>", replacement = " ") %>%
        stri_replace_all(regex="[:punct:]", replacement = " ") %>%
        trimws %>% tolower
# Corpus por lenguaje
unique(d.tm$lengua) %>%
        lapply(function(x) Corpus(VectorSource(d.tm[d.tm$lengua==x,"text"]))) -> d.corp
# Retirar stopwords
d.corp %>%
        lapply(function(x) tm_map(x,removeWords, unlist(lapply(unique(d.tm$lengua),
                stopwords)))) -> d.stop
# Stems
d.stop %>%
        lapply(function(x) tm_map(x, stemDocument, language = "en")) %>%
        lapply(function(x) tm_map(x, stemDocument, language = "es")) %>%
        lapply(function(x) tm_map(x, stemDocument, language = "de")) %>%
        lapply(function(x) tm_map(x, stemDocument, language = "fr")) %>%
        lapply(function(x) tm_map(x, stemDocument, language = "pt")) -> d.stem

# Formar matrices
d.stop %<>% lapply(DocumentTermMatrix)
d.stem %<>% lapply(DocumentTermMatrix)

# Exportar
data.table(d)[, .(tt = .N,rt = sum(rt, na.rm = TRUE),at = sum(at, na.rm = TRUE),
                  ht = sum(ht, na.rm = TRUE),
                  ht.at = sum(ht|at, na.rm = TRUE))
              , .(lengua)] -> resumen
resumen[, ht.at := ht + at - ht.at]
save(resumen, rt.df, at.df, ht.df, d.stem, d.stop, file = "02_output.Rdata")
```

	text	lengua	n.rt
	<chr>	<chr>	<int>
1	RT @betsattt1 Real Madrid - Bayern Munih	de	23
2	RT @ChampionsLeague GOAL! Bayern 1-0 Real Madrid (Joshua Kimmich 28) #U~	de	93
3	RT @nomeserrados bayern demoniaco	de	134
4	RT @ChampionsLeague GOAL! Bayern 1-2 Real Madrid (Marco Asensio 57) #UCL	en	108
5	RT @ChampionsLeague Cristiano Ronaldo <c3><b0> All eyes on this season'~	en	130
6	RT @ChampionsLeague Marco Asensio in the knockouts for Real Madrid 2016~	en	172
7	RT @ZB_Media02 Tercer penalti no senalado en contra del Real Madrid min~	es	111
8	RT @2010MisterChip BAY 1-2 RMA (FT) - El Bayern ha perdido SEIS PARTIDO~	es	135
9	RT @2010MisterChip El PSG no habia perdido en toda la temporada en casa~	es	452
10	RT @ActuFoot_ L'Allianz Arena plus comble que jamais ! Quel stade !	fr	58
11	"RT @YassEncore Real vs Bayern Ribery vs Benzema les deux \" racailles ~	fr	64
12	RT @AmineMaTue Le Real Madrid c'est incroyable tout les matchs c'est pa~	fr	110
13	RT @B24pt Enquanto a maioria v<c3><83><c2><aa> o Bayern - Real Madrid D~	pt	70
14	RT @InfosFuteboI Marco Asensio a j<c3><83><c2><b3>ia do Real Madrid! <c~	pt	123
15	RT @InfosFuteboI Quando toca o hino da Champions League	pt	157

(a) Retweets.

	at	n	lengua
	<chr>	<int>	<chr>
1	@FCBayern	59	de
2	@nomeserrados	134	de
3	@ChampionsLeague	141	de
4	@brfootball	187	en
5	@FOXSoccer	221	en
6	@ChampionsLeague	1215	en
7	@realmadrid	241	es
8	@InvictosSomos	365	es
9	@2010MisterChip	1300	es
10	@YassEncore	74	fr
11	@AmineMaTue	110	fr
12	@ActuFoot_	242	fr
13	@FoxSportsBrasil	113	pt
14	@RMadridDepre	158	pt
15	@InfosFuteboI	504	pt

(b) Usuarios.

	ht	n	lengua
	<chr>	<int>	<chr>
1	#Bayern	76	de
2	#UCL	236	de
3	#FCBRMA	288	de
4	#BAYRMA	412	en
5	#FCBRMA	484	en
6	#UCL	1505	en
7	#HalaMadrid	229	es
8	#ChampionsLeague	302	es
9	#UCL	305	es
10	#UEFAChampionsLeague	55	fr
11	#FCBRMA	141	fr
12	#BAYRMA	179	fr
13	#ChampionsLeague	45	pt
14	#UCL	78	pt
15	#HalaMadrid	104	pt

(c) Hashtags.

```
$Portuges
   bayern      real    madrid       nao cristiano
     2645      2066      1661       735       504

$Frances
   bayern      real    madrid     match champions
      812       270       108        84        67

$Ingles
bayern madrid   real munich league
  4089   3308   3047   1221   1036

$Espanol
madrid bayern   real munich    gol
  3175   3019   2740    654    402

$Aleman
 bayern    real  madrid  munich munchen
   1452     448     408     210     127
```

(d) ....

```
$Portuges
      nao cristiano   ronaldo      jogo       gol
        735       504       495       485       470

$Frances
    match champions      plus    munich    league
         84        67        60        56        44

$Ingles
   munich    league champions      live     watch
     1221      1036      1023      1018       489

$Espanol
   munich       gol   partido champions     vamos
      654       402       377       362       251

$Aleman
 munich munchen    live     fur kimmich
    210     127     126      76      66
```

(e) ....

Fig. S1. Resumen de datos

D. Análisis descriptivo y Workcloud

En esta subsección, se desarrollará un análisis descriptivo y un análisis de nube de palabras (Workcloud) de las palabras con mayor relevancia o popularidad respecto a los 5 idiomas referentes.

Listing 8. Inicialización.

```
library(ggplot2)
library(reshape2)
library(magrittr)
library(dplyr)
library(tm)
library(wordcloud)

# Cargar datos
load("02_output.Rdata")
# Prepro
resumen$tt <- with(resumen, tt - rt - at - ht - ht.at)
```

Los análisis descriptivos será evaluados sobre la frecuencia y proporciones del números de tweets respecto a cada idioma. El código y resultados son los siguientes.

Listing 9. Análisis Descriptivo.

```
# Tipos de tweet por idioma
data.frame(lengua = resumen[, 1], t(apply(resumen[, -1], 1, function(x) x/sum(x)))) %>%
        melt(id.vars = "lengua") %>% cbind(estadistico = "Proporcion") -> d1a
d1a$value <- d1a$value*(resumen[,-1] %>% apply(1, sum) %>% max)
resumen %>% melt(id.vars = "lengua") %>% cbind(estadistico = "Frecuencia") -> d1b

d1 <- rbind(d1a, d1b)

d1$lengua %<>% factor(levels = c("en", "es", "pt", "fr","de"),
                      labels = c("Ingles", "Espanol", "Portugues", "Frances", "Aleman"))
d1$variable %<>% factor(levels = c("ht", "ht.at", "at", "tt", "rt"),
                        labels = c("Hashtag", "Hashtag + Usuario", "Usuario", "Tweet solo",
                                   "Retweet"))
d1$estadistico %<>% factor(levels = c("Frecuencia", "Proporcion"))

# Plot
ggplot(d1, aes(x = lengua, y = value, fill = variable)) +
        geom_bar(stat = "identity") + xlab("Idioma") + ylab("Numero de tweets") +
        guides(fill = guide_legend(title = "Contenido de tweet")) +
        facet_grid(. ~ estadistico, scales = "free")
```

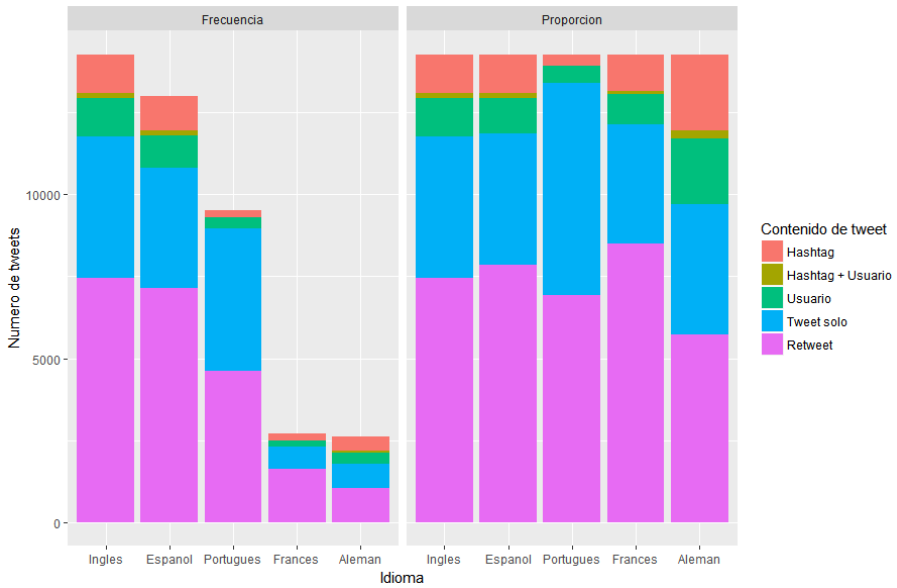


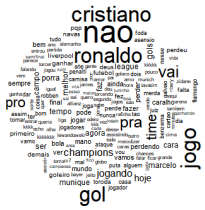
Fig. S2. Distribución de frecuencia y proporciones de los tweets por idioma.

Se diseña el código para la representación gráfica mediante WorkCloud, como se detalla a continuación

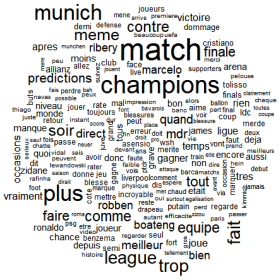
Listing 10. Análisis Workcloud.

```
# Palabras mas populares - sin stopwords
# Convertir a data.frame
d.stop %>% lapply(removeSparseTerms, 0.995) %>%
        lapply(function(x) as.data.frame(as.matrix(x))) -> d.stop.s
d.stem %>% lapply(removeSparseTerms, 0.995)  %>%
        lapply(function(x) as.data.frame(as.matrix(x))) -> d.stem.s

# Palabras mas populares, Tabla simple
d.stop.s %>%
        lapply(function(x) apply(x, 2, sum) %>%
                as.table %>%
                sort(decreasing = TRUE) %>%
                head(5)) -> d.stop.top5
names(d.stop.top5) <- c("Portuges", "Frances", "Ingles", "Espanol", "Aleman")
d.stop.top5

d.stop.s %>%
        lapply(function(x) apply(x, 2, sum) %>%
```

```
                as.table %>%
                sort %>%
                head(., length(.) - 3) %>%
                sort(decreasing = TRUE) %>%
                head(5)) -> d.stop.top5b
names(d.stop.top5b) <- c("Portuges", "Frances", "Ingles", "Espanol", "Aleman")

# Wordcloud
d.stop.s %>%
        lapply(function(x) apply(x, 2, sum) %>%
                as.table %>%
                sort %>%
                head(., length(.) - 3)) -> d.stop.ret3

for(i in 1:length(d.stop.ret3)){
        png(filename = paste0("descriptivo_04_wordcloud",i,".png"))
        wordcloud(names(d.stop.ret3[[i]]),as.vector(d.stop.ret3[[i]]), scale = c(4,.5))
        dev.off()
}
```



(a) Portugués.



(b) Francés.

(c) Inglés.

(d) Español.

(e) Alemán.

Fig. S3. Palabras más populares por cada idioma

3. ANÁLISIS DE DATOS CON TERMINO “PAOLO GUERRERO”

Los datos han sido recolectados de tweets con la frase **Paolo Guerrero** posterior al fallo del TAS del 15/05/18, organismo, que sancionó al futbolista peruano con 14 meses de suspensión. Debido a ser una información de alta relevancia en el mundo del fútbol de nuestro país y en todo el mundo se decidió a realizar un análisis de sentimientos de 1000 tweets tomados en tiempo real.

Tanto el diseño del bot que permitirá la recolección como el análisis posterior, han sido diseñados en el lenguaje de programación Python

A. Diseño del bot de recolección de datos

El mecanismo que opta el bot es el siguiente

Listing 11. Bot: Función API con autentificacion.

```
from credentials import *  Nos permite usar los codigos de acceso como variables

# Autentificacion y codigos de acceso:
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

# Retorna API con autentificacion:
api = tweepy.API(auth)
```

Posteriormente, se recolectará los tweets que contiene el término “Paolo Guerrero”, sea hashtag o simplemente una frase de tal tipo. Para ello se diseña un mecanismo de entrada, para ingresar datos respectivos, tales como el término como la cantidad de tweets a recoletar. En particular se recoletará 1000 tweets.

Listing 12. Bot: Ingreso del término en común en los tweets y cantidad a recolección.

```
Termino_buscado = input("Ingrese la palabra o hashtag a buscar:")
numero_de_Tweets=int(input("Ingrese la cantidad de tweets que desea analizar:"))

tweets = []
tweets = tweepy.Cursor(api.search,q=Termino_buscado).items(numero_de_Tweets)
```

```
Ingrese la palabra o hashtag a buscar:Paolo Guerrero
Ingrese la cantidad de tweets que desea analizar:1000
```

Fig. S4. Termino “Paolo Guerrero” en 1000 tweets recolectados.

```
RT @ExpositoresTop: He añadido un vídeo a una lista de reproducción de @YouTube (https://t.co/ZPjBJFVIVB - PAOLO GUERRERO SI
PUEDE JUGAR, L…
RT @americatv_peru: Primero habló Chilavert y ahora este crack https://t.co/dW82X4jR43
RT @ExpositoresTop: PAOLO GUERRERO SI PUEDE JUGAR, LA FIFA TIENE LA FACULTAD DE OTORGARLE AMNISTIA, #FifaAmnistiaAPaolo

¿Hay... https://t.…
RT @o_somocurcio: Aferrarnos al FIFPRO o a la campaña FIFA AMNISTIA a Paolo Guerrero no es despertar una ilusión en el hinch
a??? La FIFA no…
RT @SC_ESPN: La Federación Internacional de Futbolistas Profesionales (FIFPro) solicitó una reunión urgente con la FIFA por l
a sanción de 1…
RT @bieloficiaal: “Às pessoas que contribuíram para esta vergonhosa injustiça digo que estão me roubando o Mundial e, talvez,
minha carreir…
```

Fig. S5. Tweets Recoletados. Visualización: 6 tweets.

Es claro que debido al diseño del **Listing 4**, el estudio puede hacerse de forma general sobre cualquier término y cualquier cantidad a optar de tweets.

B. Diseño de funciones

A continuación se presentan funciones que permiten una limpieza de nuestros datos, es decir, la eliminación de caracteres innecesarios que obstaculizan el estudio; y además de una función porcentaje que permite proporcionar nuestros datos respecto al tipo de sentimientos (positivos, neutros, negativos) generados por los tweets.

Listing 13. Funciones.

```
def Eliminacion_caracteres(tweet):
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w +:\ / \ / \S +)", " ",
            tweet).split())

def porcentaje(parte,todo):
        return 100*float(parte)/float(todo)
```

C. Análisis de sentimientos (Procesamiento)

Se lleva a cabo el análisis de sentimientos generando la polaridad respectiva de cada tweet mediante la función TextBlob

Listing 14. Proceso del Análisis de Sentimientos.

```
tweetText = []

positivo = negativo = neutral = polaridad = 0

for tweet in tweets:
        print(tweet.text)
        tweetText.append(Eliminacion_caracteres(tweet.text).encode('utf-8'))
        analysis = TextBlob(tweet.text)
        polaridad += analysis.sentiment.polarity

        if (analysis.sentiment.polarity == 0.00):
                neutral +=float(1)/float(3)
                negativo +=float(1)/float(3)
                positivo +=float(1)/float(3)
        elif (analysis.sentiment.polarity < 0.00):
                negativo +=1
        elif (analysis.sentiment.polarity > 0.00):
                positivo +=1
```

D. Almacenamiento de datos recolectados

Los datos obtenidos por el bot serán almacenados en un archivo *.csv* con el objetivo de posteriores estudios.

Listing 15. Almacenamiento de datos recolectados en *.csv*.

```
import csv
csvFile = open('datos.csv', 'a')
csvWriter = csv.writer(csvFile)
csvWriter.writerow(tweetText)
csvFile.close()
```

E. Análisis de sentimientos (Resultados)

Finalmente se visualizan las proporciones respectivas por cada sentimiento tipificado.

Listing 16. Resultados.

```
positivo = porcentaje(positivo,numero_de_Tweets)
negativo = porcentaje(negativo,numero_de_Tweets)
neutral = porcentaje(neutral,numero_de_Tweets)

positivo = format(positivo,'.2f')
negativo = format(negativo,'.2f')
neutral = format(neutral,'.2f')

print("Como el publico en Twitter reacciona a " + Termino_buscado + " mediante un analisis de " +
     str(numero_de_Tweets)+ " Tweets.")

if (positivo > max(negativo,neutral)):
        print("Positivo")
elif (negativo > max(positivo,neutral)):
        print("Negativo")
elif (neutral > max(positivo,negativo)):
        print("Neutral")

labels = ['Positivo [' + str(positivo) + '%]','Neutral [' + str(neutral) + '%]','Negativo [' +
    str(negativo) + '%]']
porcentajes = [positivo,neutral,negativo]
colores = ['yellow','red','green']

patches,texts = plt.pie(porcentajes,colors = colores, startangle = 90)
plt.legend(patches,labels,loc="best")
plt.title("Como el publico en Twitter reacciona a " + Termino_buscado + " mediante un analisis de
     " + str(numero_de_Tweets)+ " Tweets.")
plt.axis('equal')
plt.tight_layout()
plt.show()
```

```
Como el publico en Twitter reacciona a Paolo Guerrero mediante un analisis de 1000 Tweets.
Negativo
```

Fig. S6. Resultado descriptivo.

Como observamos hay una tendencia negativa obtenida de la información recolectada. Debido al contexto en evaluación esto puede deberse a que la mayoría de tweets vienen relacionados con el aspecto negativo producto de la tristeza, decepción por Paolo Guerrero, además de la critica negativa hacia los agentes que han producido este clima desfavorable en el futbol peruano.

Fig. S7. Gráfico circular representativo.

Como se observa gráficamente, además de los porcentajes establecidos, que el mayor porcentaje en efecto son de sentimientos negativos; sin embargo no existe una amplia diferencia con respecto a los sentimientos positivos, estos pueden ser debido a los amplios mensajes de apoyo dedicados al futbolista. Existe además cierto porcentaje de sentimientos neutrales, estos pueden ser los relacionados con informaciones puntuales de usuarios correspondientes a noticias o los que deseen constatar informaciones en torno al tema pero sin mayor especificación de sus sentimientos.