Fecha	25/09/2015
Universidad	Tecnológico de Monterrey, Campus Santa Fe
Proyecto (Semana i)	"Business Analytics Microsoft"
Mentora	Profesora Lourdez Muñoz
Estudiantes	Gilberto Silva – Ing. en Tecnologías Computacionales Eric Zuchovicki - Ing. en Tecnologías Computacionales Alejandro Pineda – Ing. en Negocios y Tecnologías de la Información Damian Scarinci – Lic. en Administración de Empresas Alonso Alfaro – Ing. Industrial y de Sistemas

Reto: "Business Analytics Microsoft"
Reporte Técnico

ÍNDICE

TABLA DE CONTENIDOS

1.- PROCEDIMIENTO DE INSTALACIÓN Y CONFIGURACIÓN

Se requieren tener instalados los siguientes elementos:

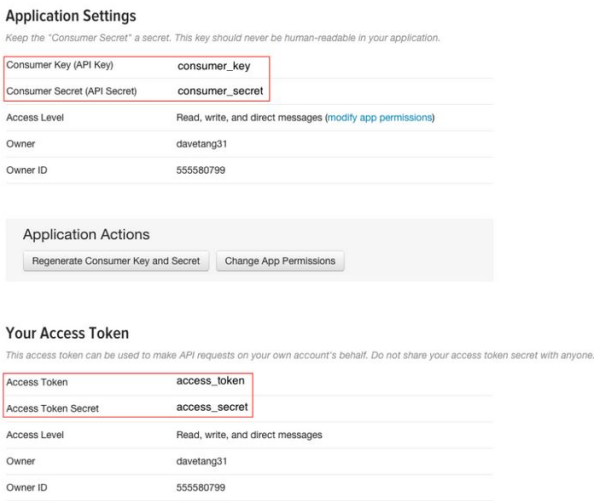
- Revolution Open
- R Studio
- Azure Machine Learning
- Power BI

Para utilizar la API de twitter, se necesitan instalar 3 paquetes en R Studio:

- twitteR: proporciona una interfaz para la API web de Twitter.
- base64enc: proporciona una codificación basada en base64. Ayuda a evitar problemas a la hora de usar twitteR.
- stringr: proporciona una fácil utilización para los strings.

Además, se necesita tener creada una aplicación en twitter, para ello:

1. Se ingresó a http://dev.twitter.com/apps y se creó una app. Para este paso, se requiere tener una cuenta de twitter, asegurándose que cuente con el número móvil registrado.
2. Se registró la aplicación. Para el registro, se pide una url, la cual sólo tiene que ser una url válida, cualquiera, por si no se tiene una propia.
3. Ya creada la app y aceptados los términos y condiciones, se tienen que configurar los permisos. Ir a la pestaña de "permisos" y darle autorización de sólo lectura.
4. Como último paso para la crear la cuenta, se deben generar las llaves. Ir a la pestaña "tokens" y generar los token de autorización. Una vez realizado esto, se necesitan las siguientes llaves (para tenerlas a la mano): consumer_key, consumer_secret, access_token, access_secret.



Para mayor información sobre la API de twitter visitar el siguiente link: https://dev.twitter.com/rest/public en donde se podrá ver más a detalle las funciones, los parámetros y limitaciones de la API.

Las demás funciones utilizada en lso script de R, no necesitan ningún paquete extra instalado. En el anexo 1, se podrán ver todos los scripts de R generados para este proyecto.

Para utilizar Azure Machine Learning, se tiene que crear una cuenta en https://**studio.azureml.net/** para poder usar todas las herramientas que te proporciona y empezar a generar modelos predictivos.

Por último, se utilizó Power BI, para generar los reportes y las gráficas, el cual no requirió de ninguna configuración especial para comenzar a usarlo.

2. – ENTENDIMIENTO DE DATOS

Comenzamos realizando algunos queries utilizando la biblioteca twitteR de R utilizando el hashtag #upgradeyourworldmx y viendo qué datos regresaba. A partir de esos datos definimos los features importantes para extraer y medir. Algunos de esos features se extraerían del texto del tweet mismo, mientras que otros se extraerían de los otros parámetros que regresaba la solicitud. La siguiente tabla ilustra los features escogidos, la razón por la cual fueron determinados importantes para medir, si fueron utilizados o no y, en caso que fueran descartados, la razón por la cual fueron descartados.

Feature	Tipo variable	Razón de elección	Utilizada	Razón de descarte
IMAGEN / FOTOGRAFÍA	Binaria	Consideramos importante saber si el tweet contaba con una foto o no. Un tweet con una foto tiene la capacidad de generar más retweets, y nos interesaba medir ese comportamiento y detectar una tendencia al respecto.	No	El api de twitter no regresa dierctamente la presencia o no de esta información, por lo que no pudimos hacer uso de ella.
VIDEO	Binaria	Consideramos importante saber si el tweet contaba con un video o no. Un tweet con una foto tiene la capacidad de generar más retweets, y nos interesaba medir ese comportamiento y detectar una tendencia al respecto.	No	El api de twitter no regresa dierctamente la presencia o no de esta información, por lo que no pudimos hacer uso de ella.

URL	Binaria	Consideramos importante saber si el tweet contaba con un url o no. Un tweet con una foto tiene la capacidad de generar más retweets, y nos interesaba medir ese comportamiento y detectar una tendencia al respecto.	No	El api de twitter no regresa dierctamente la presencia o no de esta información, por lo que no pudimos hacer uso de ella.
Timestamp de publicación	Timestamp	Consideramos que el momento en el que el tweet fue publicado puede afectar la organización por la cual se votó.	Sí	
Texto	Texto	Consideramos importante extraer el texto para de él obtener información adicional.	No	Utilizamos este dato únicamente para extraer de él más datos, pero no fue ingresado al modelo o a la graficación de información.
Screen name (Usuario)	Texto	Consideramos importante extraer el nombre de usuario para de él obtener información adicional.	No.	El api de twitter no regresa dierctamente la presencia o no de esta información, por lo que no pudimos hacer uso de ella.
Plataforma de envió	Texto	Consideramos que puede haber una relación entre la plataforma escogida y el voto realizado, y que se puede ver una tendencia.	Sí	
Tipo de voto	Texto	Consideramos importante determinar si un tweet era un tweet original, retweet de alguien más o retweet de una persona famosa (verificada).	No	No fue viable determinar para cada usuario si se trataba de un usuario verificado o no. Además, se consideró que no era importante incluir este dato

				dado que se está incluyendo la cantidad de retweets (información derivada)
Cantidad de retweets.	Numérica	Consideramos importante medir qué tan influyente fue un voto en particular, a través de la cantidad de retweets que recibió.	Sí	
Cantidad de Favoritos	Numérica	Consideramos importante saber para cuánta gente el tweet era uno de sus tweets favoritos, como otra medida del impacto que tuvo el voto.	Sí	
Followers	Numérica	Consideramos importante medir la cantidad de seguidores de un usuario al realizar un voto, ya que es una medida de qué tan influyente puede ser una publicación realizada por ese usuario.	Parcialmente	Para obtener la información de los usuarios era necesario un subquery. Debido a restricciones de tiempo y a la cantidad de queries que se puede hacer cada 15 minutos, no pudimos obtener la información de todos los usuarios que realizaron votos, sino de un subconjunto de ellos.
Amigos	Numérica	Consideramos importante medir la cantidad de personas a las que un usuario sigue al realizar un voto, ya que es una medida de qué tan activo en twitter es un usuario. Personas más activas en twitter podrían tener tendencias de voto diferentes.	Parcialmente	Para obtener la información de los usuarios era necesario un subquery. Debido a restricciones de tiempo y a la cantidad de queries que se puede hacer cada 15 minutos,

				no pudimos obtener la información de todos los usuarios que realizaron votos, sino de un subconjunto de ellos.
Cuenta Verificada	Binaria	Consideramos importante medir si un votante es un usuario verificado o no, ya que es una medida de qué tan influyente puede ser una publicación realizada por ese usuario. Usuarios verificados (famosos) pueden tener tendencias de voto diferentes y pueden tener un gran impacto en una fundación.	Parcialmente	Para obtener la información de los usuarios era necesario un subquery. Debido a restricciones de tiempo y a la cantidad de queries que se puede hacer cada 15 minutos, no pudimos obtener la información de todos los usuarios que realizaron votos, sino de un subconjunto de ellos.
palabraNinos palabraCancer palabraDiscapacidad palabraAuditiva palabraAnimal palabraLeyes palabraJoven palabraCalle palabraMedioAmbiente palabraArte palabraParalisis palabraCerebral palabraSeguridad palabraTransplante palabraIgualdad palabraRescate	Binaria	Consideramos importante medir la presencia o no de ciertas palabras en el texto de un tweet. Esto es útil para poder, en conjunto con el resto de los features, predecir por qué organización votaría una persona que quisiera votar a algún tipo de organización. Estos features tendrán un gran peso, pero no serán los únicos. La información de estas tendencias puede ayudar a fundaciones a saber en qué gente debería enfocar sus esfuerzos de difusión: gente que quiere apoyar a organizaciones con sus características pero que acabó votando por otra organización.	Sí	

palabraAlzheimer palabraHambre palabraPobreza palabraEducacion palabraVivienda palabraPeriodista palabraIndigena palabraMujer palabraViolencia palabraDeporte palabraSalud				

3. – PREPARACIÓN DE DATOS

La preparación de datos fue fundamental en el proceso ya que se realizó el análisis de qué se le necesitaba hacer a los tweets que se estaban recuperaban para poder tener resultados más precisos. Para la preparación de los datos, se utilizó R-Studio; los scripts se pueden ver en el Anexo 1. Se hizo un proceso ETL(Extract, Transform, Load).

Para el **E**xtract, mediante el uso de la API de Twitter y el software R-Studio se obtuvieron los *tweets* que incluyeran en su texto #UpgradeYourWorldMx. Se utilizó la función "searchwitter" para la descarga de los datos.

Para el **T**ransform, utilizando el software R-Studio se decidió hacer una limpieza y transformación de los datos obtenidos previamente para poder así generar un documento ordenado y listo para ser utilizado en el software Azure M.L.

Las funciones que se llevaron a cabo para preparar los datos son las siguienes:

- Eliminación de las columnas innecesarias que se obtenían con la petición de búsqueda a la API de twitter. Se descartaron las columnas:
  - id
  - favorited
  - replyToSN
  - truncated
  - replyToSID
  - replyToSN
  - replyToUID
  - longitude
  - latitude
  - isRetweet
  - retweeted
- El campo statusSource se hizo legible, es decir, se extrajo sólo la plataforma.

- Se extrajo el voto del texto del tweet, para esto se realizaron las siguientes validaciones:

    - El “voto por” se tomó en cuenta usando expresiones regulares para considerar todos los casos. Ya sea con el hashtag #votopor, o por la palabra votopor, tomándose en cuenta que podía haber mayúsculas, espacios. También, el “voto por” puede ir al final del tweet, intermedio o al final. Además, tomamos en cuenta aquellos @screenName que tenían acentos, aunque twitter no los reconozca como cuenta.
    - Una vez obtenido, el “voto por”, extraíamos el @screenName próximo, y no necesariamente justo al lado.
    - Si se encontraba un tweetcon el caracter especial “...”, se analizaba primero si existía un voto válido, y si no, lo pusimos como “VOTO INVALIDO”. Esta parte fue necesaria ya que la API de twitter, en los RT largos, lo que hace es truncarlos y no se podía alcanzar a obtener el voto.
    - Si no había un “voto por”, también se ponía como “VOTO INVALIDO”
- Una vez obtenidas las fundaciones por las cuales había votado cada usuario, se eliminaron los votos inválidos.
- Todas los valores de las filas de la columna “voto por” se pusieron en minúsculas.
- Todas los valores de las filas de la columna “voto por” que contenían acentos, se les sustituyeron los acentos por vocales.
- Sólo se contó un voto de aquellas personas que votaron dos o más veces en un día, por lo que se elimnaron los sobrantes.

* Es importante mencionar que sólo se tomaron 20, 000 tuits por día, y que nuestro resultado final, ya con la limpieza, fue un dataset de 46, 000 datos.

Finalmente para el **L**oad, se juntaron todos los archivos CSV y se hizo un documeno con todos los datos ya limpios, dado que la mayor parte del trabajo fue realizado en el software R-Studio, el documento estaba listo para ser cargado en el programa Azure M.L.

Por otra parte, se hizo un script para obtener datos de los usuarios. En esta parte del proyecto nos encontramos con un problema, ya que twitter te limita a hacer 180 requests por cuenta en una ventana de 15 minutos.

Lo que se hizo fue agarrar el archivo CSV final, y se borraron los usuarios repetidos. Con este CSV sacado del dataset final, se fueron haciendo los requests por usuario, y obteniendo su información.

En esta sección, por el tiempo que se tuvo, y las limitaciones de twitter, sólo se pudieron obtener los datos de 10,700 usuarios, y no de los 33,00 que se tenían. Se hizo un join de las tablas y se obtuvo un dataset final con 25, 000 registros, ya con datos de usuarios.

4. – MODELADO

Finalmente, se decidio hacer dos modelos predictivos distintos. El primer modelo cuenta únicamente con las características del tweet, tales como: el numero de veces que fue retweeteado, el número de veces que fue marcado como favorito, la plataforma por la que fue envíado; etc. El segundo modelo, a demás de incluir todas las variables del primer modelo, incluyo las características del usuario que envía el tweet, características como: si el usuario esta verificado o no, la cantidad de seguidores del usuario; etc.

Ambos modelos fueron definidos como un problema de clasificación de muchas clases. El resultado que arroja el modelo predictivo es por qué fundación votará un usuario dados todos los parámetros mencionados. Cada una de las fundaciones es una posible clasificación y el modelo calculará la probabilidad de que el voto vaya dirigido a cada una de esas fundaciones.

A continuación se muestran las variables tomadas en cuenta para cada modelo.

Modelo 1

Y1	X1	X2	X3	X4	X5...X32
Fundación	**Recuento de Fav's**	**Recuento de RT's**	**Fecha**	**Plataforma de Envío**	**Palabras Clave**

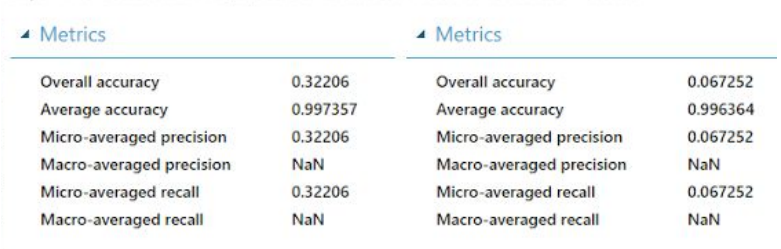
Modelo 2

Y1	X1	X2	X3	X4	X5	X6	X7	X8...X35
Fundación	**# Fav's**	**# RT's**	**Fecha**	**Plataforma**	**Seguidores del usuario**	**Amigos del usuario**	**Verificado**	**Palabras Clave**

El modelo fue realizado en Azure Machine Learning.

1. Subimos los archivos CSV a la plataforma (uno con los datos de cada modelo). El modelo 1 contó con una mayor cantidad de datos, mientras que el modelo 2 contó con una mayor cantidad de features (los del usuario).
2. Realizamos el filtrado de las columnas que no utilizaríamos, que fueron únicamente columnas en blanco, el texto del tweet y el nombre de usuario, ya que el resto de los datos inútiles fueron limpiados en los pasos anteriores.
3. Separamos los datos en conjunto de entrenamiento (80%) y conjunto de prueba (20%).
4. Entrenamos el modelo con el método de clasificación escogido. Probamos una Red Neuronal, Decision Forest y Decision Jungle, y medimos la precisión de cada uno de

ellos. El modelo que acabó teniendo un mejor resultado fue el Decision Forest, y fue el que utilizamos.



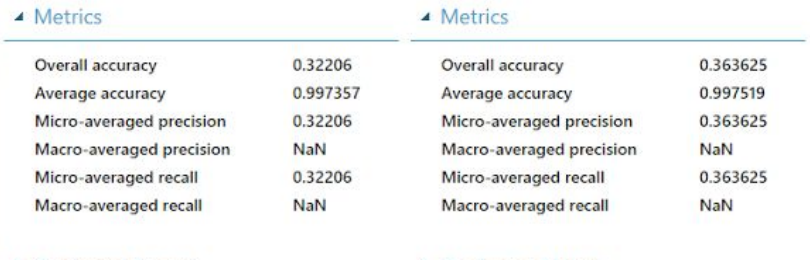
Metrics

Overall accuracy	0.32206
Average accuracy	0.997357
Micro-averaged precision	0.32206
Macro-averaged precision	NaN
Micro-averaged recall	0.32206
Macro-averaged recall	NaN

Metrics

Overall accuracy	0.067252
Average accuracy	0.996364
Micro-averaged precision	0.067252
Macro-averaged precision	NaN
Micro-averaged recall	0.067252
Macro-averaged recall	NaN

Precisión de Decision Jungle (izquierda) vs Red Neuronal (derecha)



Metrics

Overall accuracy	0.32206
Average accuracy	0.997357
Micro-averaged precision	0.32206
Macro-averaged precision	NaN
Micro-averaged recall	0.32206
Macro-averaged recall	NaN

Metrics

Overall accuracy	0.363625
Average accuracy	0.997519
Micro-averaged precision	0.363625
Macro-averaged precision	NaN
Micro-averaged recall	0.363625
Macro-averaged recall	NaN

Precisión de Decision Jungle (izquierda) vs Decision Forest (derecha)

5. Calificamos y evaluamos el modelo con el conjunto de prueba.

Modelo final resultante

6. A partir del modelo generamos una aplicación web predictiva. De esa manera descargamos el archivo de excel generado que contiene un macro que, al ser llenado, realiza una solicitud al modelo, enviándole los datos, y obtiene de regreso las probabilidades de votar por cada organización así como la organización con la mayor probabilidad.

5.- EVALUACIÓN

Como se mencionó anteriormente, debido a que el modelo Decision Forest tuvo un mejor desempeño (comparado con Decision Jungle y Red Neuronal), optamos por utilizarlo. La métrica a la que se le dio una mayor importancia de las anteriores fue la precisión promedio, que es de 99.75. Además, pudimos ver la matriz de confusión (cuyo tamaño es demasiado grande para poner en reporte o pantalla, debido a que tenemos una gran cantidad de categorías) pudimos observar que gran cantidad de las predicciones las realizaba correctamente.

Métricas

Además, en los macros de excel generados (explicados a continuación) realizamos varias pruebas empíricas y verificamos que los resultados arrojados eran correctos o razonables.

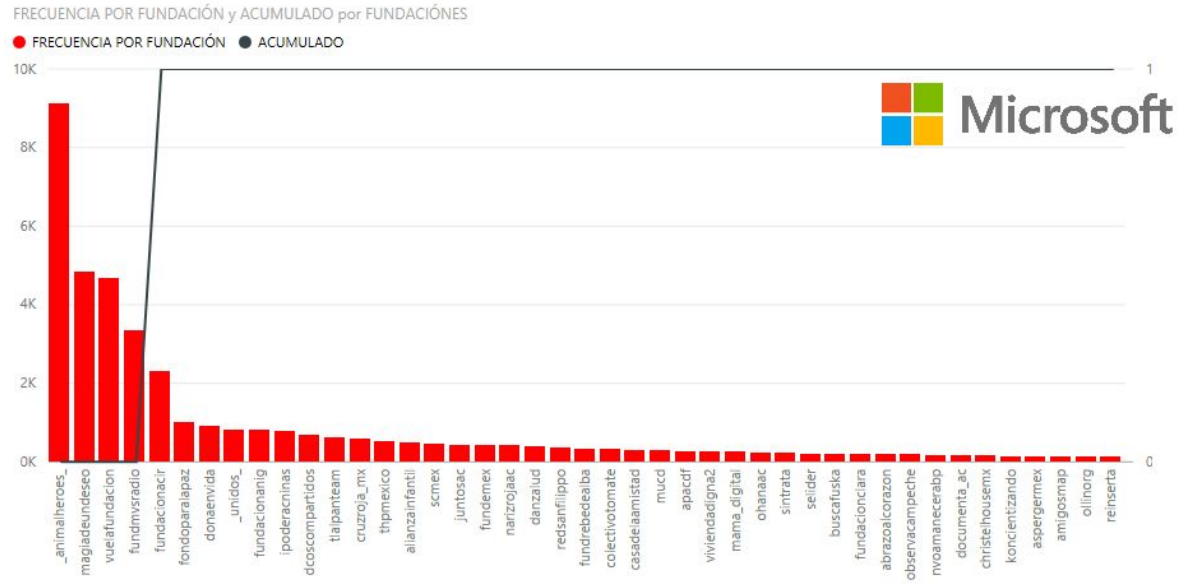
6.- DESPLIEGUE

Como endpoint de cada uno de los modelos generamos en Azure Machine Learning un macro de excel que cuenta con un URL y llave de la petición que debe realizar y que, al ingresar los features de entrada, manda la solicitud y obtiene de regreso cuál es la probabilidad de que ese voto vaya destinado a cada una de las organizaciones, así como cuál es la fundación con mayor probabilidad. Las columnas de las variables que no son utilizadas como entrada al modelo son ignoradas.

Intentamos realizar una aplicación web para realizar dichas solicitudes de una mejor manera a través de un formulario html. Creamos la aplicación web en www.semanai.azurewebsites.net con esa intención y escribimos el formulario requerido. Sin

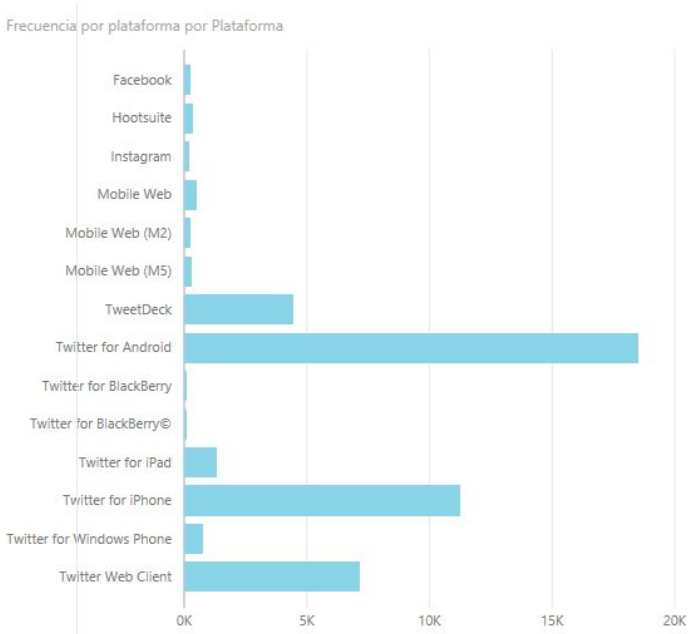
embargo, no logramos configurar el servicio web de la aplicación que se comunicara con el servicio web de Machine Learning, y por cuestiones de tiempo decidimos abandonarla.

Una vez recolectados los datos y creado el modelo predictivo, se optó por modelar los resultados obtenidos en nuestro modelo y de la misma manera una especulación, acerca de qué hubiera sucedido si se hubiera tomado en cuenta estrictamente los votos efectuados de la manera en la que se pidió en la campaña.
Primeramente se graficó, utilizando un diagrama de Pareto, en el cual se muestran las fundaciones en su totalidad y se muestra donde recaen la mayoría de los votos, en este caso, las fundaciones más sobresalientes fueron: _animalheroes_, magiadeundeseo, vuelafundacion, fundacionmvsradio y fundacionzcir, en las cuales recaen casi todos los datos, como se muestra a continuación:



Dentro de la siguiente gráfica, con su respectiva tabla, se filtraron los resultados de las plataformas que fueron más utilizadas para generar los votos para las fundaciones, cabe recalcar que muchas de estas plataformas fueron eliminadas, ya que formaban parte de menos del 0.1% de los votos ya que solamente uno o dos votos fueron efectuados sobre esta plataforma, siendo predominantes las plataformas de Android, iPhone, Web, y TweetDeck. La gráfica se muestra a continuación:

Plataforma	Frecuencia por plataforma
Twitter for Android	18528
Twitter for iPhone	11254
Twitter Web Client	7164
TweetDeck	4434
Twitter for iPad	1349
Twitter for Windows Phone	779
Mobile Web	495
Hootsuite	334
Mobile Web (M5)	310
Mobile Web (M2)	275
Facebook	231
Instagram	197
Twitter for BlackBerry©	122
Twitter for BlackBerry	114
Total	**45586**



Por último se crearon dos gráficas, la primera mostrando un modelo ideal, en el cual se tomaban solamente y estrictamente los votos que fueron redactados de manera correcta (como se pidió que se escribieran los votos en la convocatoria), y por otra parte, se tomaron los votos como sucedió en realidad, se tomaron en cuenta los votos con cierta holgura, tomando en cuenta faltas de ortografía y hashtags escritos de manera irregular, las tablas con los contadores se muestran junto con las gráficas. Cabe recalcar que los resultados, principales no varían, pero algunas de las asociaciones dentro del top 10 si cambian de lugar debido a esta restricción de votos. Estas instancias se muestran a continuación:

Top 10 fundaciones Estrictas	Votos Estrictos
animalheroes	7176
magiadeundeseo	4395
vuelafundacion	3181
fundmvsradio	1631
fundacionacir	1557
unidos	575
donaenvida	570
fondoparalapaz	467
cruzroja_mx	435
alianzainfantil	424
Total	**20411**

Top 10 fundaciones con Holgura	Votos con Holgura
animalheroes	9133
magiadeundeseo	4831
vuelafundacion	4681
fundmvsradio	3354
fundacionacir	2305
fondoparalapaz	1008
donaenvida	914
unidos	827
fundacionanig	815
ipoderacninas	780
Total	**28648**

Estas gráficas muestran que, dado que se hubiera tomado en cuenta solamente los votos estrictamente bien escritos, los resultados hubieran mostrado un ligero cambio de lugares entre ciertas fundaciones.

Estas graficas no son las únicas que se tomaron y consideraron pero si fueron las más influyentes, para mayor información acerca de las gráficas restantes, favor de dirigirse al proyecto de Power BI anexado en la entrega final.

7.- REFERENCIAS

- http://davetang.org/muse/2013/04/06/using-the-r_twitter-package/
- http://www.endmemo.com/program/R/grepl.php
- http://www.endmemo.com/program/R/gsub.php
- http://rfunction.com/archives/2354
- http://code.hammerpig.com/loops-continue.html

8.- ANEXOS

Anexo 1 - Script 1

```
#########################################################################
# Semana I
#
# Script en R para extraer los tweets de un día
#
#########################################################################
#
# Equipo 5
# Integrantes:
## Gilberto Silva – Ing. en Tecnologías Computacionales
## Eric Zuchovicki - Ing. en Tecnologías Computacionales
## Alejandro Pineda – Ing. en Negocios y Tecnologías de la Información
## Damian Scarinci – Lic. en Administración de Empresas
## Alonso Alfaro – Ing. Industrial y de Sistemas

# Verificar paquetes
if(!require("twitteR")){install.packages("twitteR")}
if(!require("base64enc")){install.packages("base64enc")}
if(!require("stringr")){install.packages("stringr")}

# Inicializar la API
init_cred <- function(){
  twitter_cred <- list()
  twitter_cred$consumer_key <- "key_here"
  twitter_cred$consumer_secret <- "key_here"
  twitter_cred$access_token <- "key_here"
  twitter_cred$access_secret <- "key_here"
  write.csv(twitter_cred, file="C:/Users/user_here/Documents/twitter_cred.csv")
```

```
}

twitter_cred <- read.csv("twitter_cred.csv", stringsAsFactors = FALSE)
setup_twitter_oauth(twitter_cred$consumer_key, twitter_cred$consumer_secret,
twitter_cred$access_token, twitter_cred$access_secret)
ds <- searchTwitter("#upgradeyourworldmx",n=20000, since="2015-09-19",
until="2015-09-20", locale = NULL)#,  since = "2015-09-01", until = "2015-09-02")

# Definir dataFrame
df1 <- twListToDF(ds)   # Comentar si no se extraen tweets

############
# Leer de un csv en vez de extraer tweets
#test = read.csv("regular_expressions.csv")
#df1 <- test
###########

# obtener el texto del tweet con su respectiva columna
tweetText <- df1[,c("text")]

#################################################################################
#####################
##### Expresiones Regulares

# Pobreza
df1$palabraPobreza <- grepl("pobre[(za)]?",tweetText, ignore.case = TRUE) + 0
# Niño / Niña
df1$palabraNinos <- grepl("niñ[oa]s?",tweetText, ignore.case = TRUE) + 0
# Cáncer
df1$palabraCancer <- grepl("c[aá]ncer",tweetText, ignore.case = TRUE) + 0
# Discapacidad
df1$palabraDiscapacidad <- grepl("discapaci",tweetText, ignore.case = TRUE) + 0
# Auditiva
df1$palabraAuditiva <- grepl("auditiv[ao]?|auditividad|escuch",tweetText, ignore.case =
TRUE) + 0
# Animal
df1$palabraAnimal <- grepl("animal[(es)]?",tweetText, ignore.case = TRUE) + 0
# Leyes
df1$palabraLeyes <- grepl("ley[(es)]?",tweetText, ignore.case = TRUE) + 0
# Joven
df1$palabraJoven <- grepl("j[óo]ven[(es)]?|juventud",tweetText, ignore.case = TRUE) + 0
# Calle
df1$palabraCalle <- grepl("calle|callejer[oa]?[s]*",tweetText, ignore.case = TRUE) + 0
# Medio Ambiente
```

```
df1$palabraMedioAmbiente <- grepl("medio(\\s)?ambiente|ecol[oó]g[ií]",tweetText, ignore.case = TRUE) + 0
# Arte
df1$palabraArte <- grepl("art[e(ista)]?[s]?",tweetText, ignore.case = TRUE) + 0
# Paralisis
df1$palabraParalisis <- grepl("par[aá]lisis",tweetText, ignore.case = TRUE) + 0
# Cerebral
df1$palabraCerebral <- grepl("cerebr(al|o)",tweetText, ignore.case = TRUE) + 0
# Seguridad
df1$palabraSeguridad <- grepl("segur(o|a|(idad)])?[s]?",tweetText, ignore.case = TRUE) + 0
# Transplante
df1$palabraTransplante <- grepl("tra[n]?splante[s]?",tweetText, ignore.case = TRUE) + 0
# Igualdad
df1$palabraIgualdad <- grepl("igualdad|equidad",tweetText, ignore.case = TRUE) + 0
# Rescate
df1$palabraRescate <- grepl("rescat[e|(ar)]?[s]?",tweetText, ignore.case = TRUE) + 0
# Alzheimer
df1$palabraAlzheimer <- grepl("alzheimer",tweetText, ignore.case = TRUE) + 0
# Hambre
df1$palabraHambre <- grepl("hambr([e|(iento)|(ienta)])|alimenta([r|(ci[oó]n)])|hambruna",tweetText, ignore.case = TRUE) + 0
# EducaciÃ³n
df1$palabraEducacion <- grepl("educa((ci[oó]n)?[r]?)?|enseña[(nza)|r]?|apendizaje",tweetText, ignore.case = TRUE) + 0
# Vivienda
df1$palabraVivienda <- grepl("vivienda[s]?",tweetText, ignore.case = TRUE) + 0
# Periodista
df1$palabraPeriodista <- grepl("periodis[(mo)]?|periodis[(ta)[s]?)]?",tweetText, ignore.case = TRUE) + 0
# Indígena
df1$palabraIndigena <- grepl("ind[ií]gena[s]?",tweetText, ignore.case = TRUE) + 0
# Mujer
df1$palabraMujer <- grepl("mujer[es]?",tweetText, ignore.case = TRUE) + 0
# Violencia
df1$palabraViolencia <- grepl("violen[(to)|(ta)|(cia)]?[s]?",tweetText, ignore.case = TRUE) + 0
# Deporte
df1$palabraDeporte <- grepl("deport[e(ista)][s]?",tweetText, ignore.case = TRUE) + 0
# Salud
df1$palabraSalud <- grepl("salud((\\s)|($))",tweetText, ignore.case = TRUE) + 0

##############################################################################
############################
```

```
###### Limpieza de datos

# Remover columnas innecesarias
df1$id <- NULL
df1$favorited <- NULL
df1$replyToSN <- NULL
df1$truncated <- NULL
df1$replyToSID <- NULL
df1$replyToSN <- NULL
df1$replyToUID <- NULL
df1$longitude <- NULL
df1$latitude <- NULL
df1$isRetweet <- NULL
df1$retweeted <- NULL

# Extraer plataforma (puro string)
platformString <- df1[,c("statusSource")]
df1$statusSource <- gsub(".*\">", "", platformString)
platformString2 <- df1[,c("statusSource")]
df1$statusSource <- gsub("</a>$", "", platformString2)

# Obtener el voto y limpiar los invalidos
df1$votoPor <- sub("^.*@\\w+\x85$", "VOTO INVALIDO",tweetText, ignore.case = TRUE, perl = TRUE)

#Considera los casos en los que #votopor esta inmediatamente seguido por la fundacion
df1$votoPor <- sub("^.*#votopor\\s*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#votopor\\s*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#voto\\s*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#voto\\s*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovotopor\\s*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovotopor\\s*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*vot[eo]\\s?por\\s+@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*vot[eo]\\s?por\\s+@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovoto\\s*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovoto\\s*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#votapor\\s*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#votapor\\s*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
```

```
#Considera los casos en los que #votopor no esta inmediatamente seguido por la fundacion
df1$votoPor <- sub("^.*#votopor\\s*.*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#votopor\\s*.*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*@(\\w+)\\W.*#votopor\\s*.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#voto\\s*.*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#voto\\s*.*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovotopor\\s*.*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovotopor\\s*.*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*vot[eo]\\s?por\\s+.*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*vot[eo]\\s?por\\s+.*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovoto\\s*.*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#yovoto\\s*.*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#votapor\\s*.*@(\\w+)\\W.*$", "\\1",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("^.*#votapor\\s*.*@(\\w+)$", "\\1",df1$votoPor, ignore.case = TRUE)

# Eliminar los votos inválidos
df1 <- df1[!df1$votoPor == "VOTO INVALIDO", ]

# Pasar a minúsculas la columna voto por
fundacionLower <- df1[,c("votoPor")]
df1$votoPor <- tolower(fundacionLower)

# Sustituye los acentos
df1$votoPor <- sub("á", "a",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("é", "e",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("í", "i",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("ó", "o",df1$votoPor, ignore.case = TRUE)
df1$votoPor <- sub("ú", "u",df1$votoPor, ignore.case = TRUE)

# Eliminar duplicados
user <- df1[,c("screenName")]
df1 <- df1[!duplicated(user), ]

################################################################################
############################
##### Escribimos en un CSV el resultado final del día
```

```
write.csv(df1, file=paste(getwd(),"/csv_dia_sep.csv",sep=""))
```

Anexo 2 - Script 2:

```
#########################################################################
# Semana I
#
# Script en R para extraer la información de un usuario
#
#########################################################################
#
# Equipo 5
# Integrantes:
## Gilberto Silva – Ing. en Tecnologías Computacionales
## Eric Zuchovicki - Ing. en Tecnologías Computacionales
## Alejandro Pineda – Ing. en Negocios y Tecnologías de la Información
## Damian Scarinci – Lic. en Administración de Empresas
## Alonso Alfaro – Ing. Industrial y de Sistemas

if(!require("twitteR")){install.packages("twitteR")}
if(!require("base64enc")){install.packages("base64enc")}
if(!require("stringr")){install.packages("stringr")}

######################
# Leer CSV con todos los usuarios
csvFile = read.csv("dataset_final_users.csv", stringsAsFactors = FALSE)
dfus <- csvFile

# Se cambiarán las keys dinámicamente por lo que se pone en false
options(httr_oauth_cache=F)

###########################################

# Keys gil (primer integrante)
api_key <- "key_here"
api_secret <- "key_here"
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 1:180) {
  tryCatch({
        tuser <-getUser(dfus$screenName[i])  # obtiene el usuario
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
```

```
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
        next
  })
}

################################################################################
################
#Cambio de keys y vuelvo a ejcutar

# Keys alex (segundo integrante)
api_key <- "key_here"
api_secret <- "key_here"
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 181:360) {
  tryCatch({
        tuser <-getUser(dfus$screenName[i])
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
        next
  })
}

################################################################################
################
#Cambio de keys y vuelvo a ejcutar

# Keys erick (tercer integrante)
api_key <- "key_here"
api_secret <- "key_here"
```

```
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 361:540) {
  tryCatch({
        tuser <-getUser(dfus$screenName[i])
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
        next
  })
}

################################################################################
################
#Cambio de keys y vuelvo a ejcutar

# Keys alonso (cuarto integrante)
api_key <- "key_here"
api_secret <- "key_here"
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 541:720) {
  tryCatch({
        tuser <-getUser(dfus$screenName[i])
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
```

```
        next
  })
}

##############################################################################
################
#Cambio de keys y vuelvo a ejcutar

# Keys damian (sexto integrante)
api_key <- "key_here"
api_secret <- "key_here"
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 721:900) {
  tryCatch({
        tuser <-getUser(dfus$screenName[i])
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
        next
  })
}

##############################################################################
################
#Cambio de keys y vuelvo a ejcutar

# Keys pato (cuenta externa)
api_key <- "key_here"
api_secret <- "key_here"
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 901:1080) {
  tryCatch({
```

```
        tuser <-getUser(dfus$screenName[i])
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
        next
  })
}

################################################################################
################
#Cambio de keys y vuelvo a ejcutar

# Keys laura (cuenta externa)
api_key <- "key_here"
api_secret <- "key_here"
access_token <- "key_here"
access_token_secret <- "key_here"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

for (i in 1081:1260) {
  tryCatch({
        tuser <-getUser(dfus$screenName[i])
        dfus$followersCount[i] <- tuser$followersCount
        dfus$verified[i] <- tuser$verified + 0
        dfus$location[i] <- tuser$location
        dfus$friendsCount[i] <- tuser$friendsCount
        dfus$protected[i] <- tuser$protected + 0
  }, warning = function(w) {
        print(w)
  }, error = function(e) {
        print(e)
  }, finally = {
        next
  })
}

################################################################################
############################
```

```
##### Escribimos en un CSV
write.csv(dfus, file=paste(getwd(),"/dataset_final_users.csv",sep=""))
```