ITAM Análisis clickstream

Ciencia de Datos

4 de noviembre de 2015

En un sitio Web, el análisis de *"clicksstream"* es el proceso de recolección, análisis y presentación de datos agregados sobre cada paso que siguen los visitantes en una página web y en qué orden, es decir, son el resultado de la sucesión de clicks del ratón por cada visitante. Regularmente, dicho flujo o registro de información es almacenado en un principio para la gestión de los registros y posteriormente son analizados para producir estadísticas que resulten de utilidad.

0. El problema por resolver

Para realizar dicho análisis es necesario obtener y estudiar los datos provenientes de los *access.logs* principalmente porque son datos no estructurados. Una vez estructurados, se necesitan quitar los registros duplicados, sesionizar los datos y enriquecer los registros ya que debido a las características que aporta cada registro, suelen no ser suficientes para tener un análisis más detallado.

Cabe destacar que no sólo la resolución del problema llega hasta esta fase, sino hasta la creación de un dashboard o tablero de estadísticas que ayudan a visualizar las principales características de los datos; por ejemplo, los usuarios que tienen más visitas, su tiempo de permanencia, las páginas más visitadas, entre otras.

1. Orquestación

Al igual que en el análsis de Text Miner, se optó por utilizar el orquestador luigi, pero para el análisis de *clickstream*, las ventajas que se utilizan son las de *modularidad*, *robustez* e *idempotencia*. Cabe resaltar que para esta fase del proyecto, debido a que no se tienen archivos access.log provenientes de *D-space*, los archivos logs fueron obtenidos de diversas fuentes que serán mencionadas más adelante.

A continuación se describe los pasos del *pipeline* que son ejecutados por la función principal *analisis-log-itam.py*; en cada uno de estos se explica su función, así como también el archivo *input* y *output* necesario.

(i) Inputlog

Es el primer paso del pipeline y es el encargado de importar el archivo *access.logs* de la ruta predeterminada hacia el orquestador (*luigi*).

Como antecedente, este proceso se ejecutaba por medio de *batch* en el lenguaje de programación *pearl* por medio de la función *accesslog2csv.pl*; sin embargo, se decidió integrar este paso a la orquestación de *luigi* para que el proceso se ejecutado en una sola orquestación.

input: access.log

output: inputlog.pd (archivo data frame pandas)

(ii) Parsear

Después de que *luigi* recibe el access.log,este paso es el encargado de nombrar las variables y la estructura del archivo *access.log*. La estructura y los nombres de las variables fueron tomadas de los *CustomLog* apache.org.

Las variables que se tienen son las siguientes:

- Host
- Log_Name
- Date_Time
- Method
- Response_Code
- Bytes_Sent
- URL
- User_Agent

input: inputlog.pd

output: parsear.pd

(iii) Usuario

En este paso se necesita tener identificados a los usuarios en sentido en la forma en que visitaran la página de *D-space* ya que se el servicio de consulta de la biblioteca de arte, puede llevarse a cabo por medio de una computadora por usuario o varios usuarios en una misma computadora.

Para poder realizar este paso, nos basamos en el supuesto que un usuario, solo visitará el sitio por medio de una computadora.

input: parsear.pd

output: usuario.pd

Este paso es posible que sea modificado ya definida la estructura de consulta de *D-space*. Dentro de la función *analisis-log-itam.py* se tiene comentado en dónde se realizaría dicho cambio.

(iv) Sesionizar

La función de este paso es ordenar los registros por fecha y usuario, quitar duplicados y agregar 2 campos nuevos que son:

- time_diff: calcula la diferencia en tiempo entre una consultas del usuario en la página *web* siendo el primer registro puesto como *0* ya que no se tiene contra quien comparar.
- Rank: crea un "ranking" entre las consultas por usuario, siendo *1* la primera consulta del usuario, *2* la segunda y así sucesivamente.

input: usuario.pd

output: sesionizar.pd

(v) Enriquecer

En este paso se crean nuevas variables o campos que podrían ser de interés para el análisis. Los campos que se crean por consulta son los siguientes:

- year: año
- month: mes
- day: dia (número)
- hour: hora

- day of week: día de la semana
- dif seg clicks: segundos de consulta

En este paso, pueden ser agregados más campos que sean de interés para los administradores del sistema. Los campos descritos con anterioridad son los más comunes ya que nos ayudan a determinar los días, horas, mes y años de visitas más o menos frecuentes y/o el tiempo promedio de permanencia.

El output de tipo *csv* es el insumo principal del dashboard.

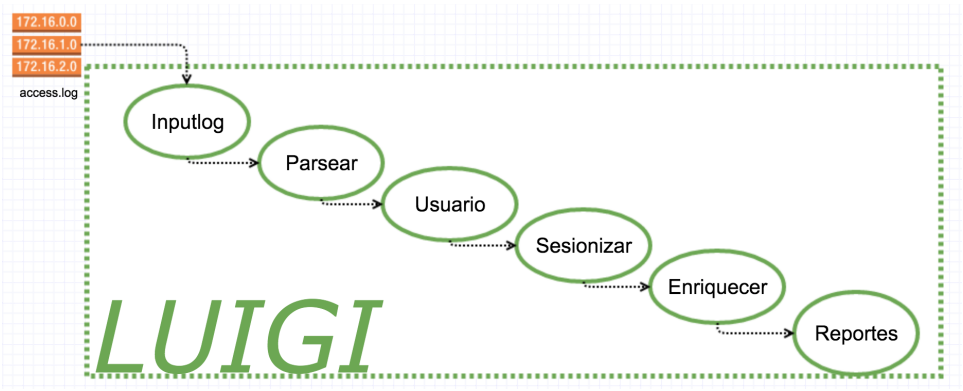
input: sesionizar.pd output: enriquecer.csv o enriquecer.pd

(vi) Reportes

De manera automática son creados reportes en *pdf* de los *códigos de respuesta* y *usuarios* con las estadísticas del porcentaje de visitas y el tiempo promedio de consulta.

input: enriquecer.pd

output: reportes.pdf



2. Instalación

Para la ejecución del pipeline es necesario instalar `Python`, `R` y bibliotecas de `Python` usando `apt-get`

```
# Instalar R

    sudo apt-get update
    sudo apt-get install r-base

# Instalar dependencias de R

En la línea de comando para verificar la correcta instalación teclaer R

Entraras a la línea de comandos de R, teclear:

  install.packages("shiny")
  install.packages("dplyr")
  install.packages("ggplot2")

Una vez instalado las librerias, salir de R
```

```
  quit()

# Instalar Python

Volveras a la línea de comandos, teclear:

sudo apt-get install build-essential checkinstall
sudo apt-get install libreadline-gplv2-dev libncursesw5-dev \
libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev

sudo apt-get install python2.7

# Instalar dependencias de Python

  sudo pip install luigi
  sudo apt-get install python-matplotlib
```

3 Breve guía de uso

Una vez instalado lo anterior, se necesitan 3 cosas para ejecutar el pipeline correctamente, la función *analisis-log-itam.py*, el archivo *access.log* y la carpeta de *functions* la cuál trae funciones externas. Lo necesario deberá estar en una ruta especificada por el usuario.

En la línea de comandos ir hasta la ruta donde se tiene lo anterior

```
Ejemplo:
  cd User/miruta/misarchivos
```

Para ejecutar el pipeline y visualizar el proceso es necesario abrir otra línea de comandos y el navegador

```
Dentro de la línea de comandos teclear:

  luigid

Dentro del navegador, abrir el puerto y poner la siguiente dirección:

  http://localhost:8082/static/visualiser/index.html#
```

Posteriormente en la línea de comandos (diferente a donde se tecleo *luigid*), ejecutar la función.

```
python analisis-log-itam.py Enriquecer --input-file access.log --output-file\
usuario.pd  --output-df sesionizar.pd --output-df1 enriquecer --output-df2\
reporte.pd
```

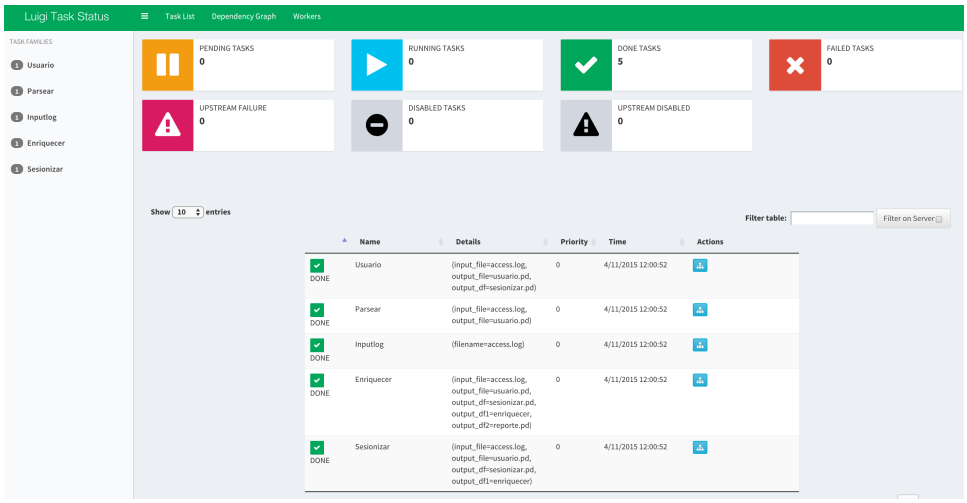
La salida que se obtendrá será la siguiente

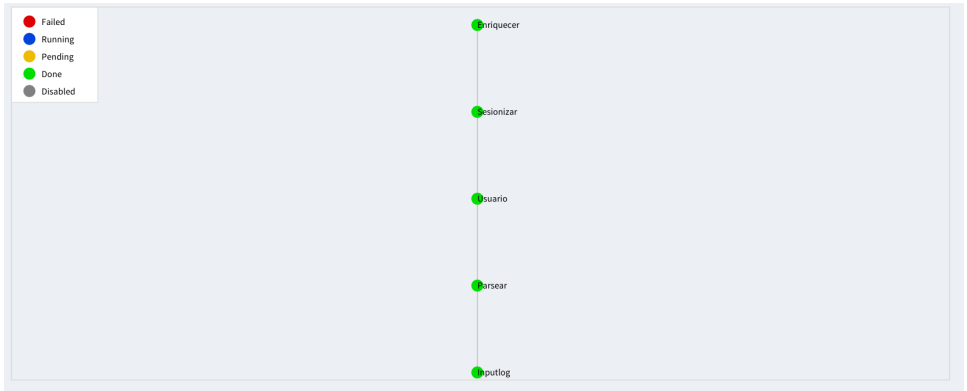
```
===== Luigi Execution Summary =====

Scheduled 5 tasks of which:
* 1 present dependencies were encountered:
```

```
    - 1 Inputlog(filename=access.log)

* 4 ran successfully:

    - 1 Sesionizar(input_file=access.log, output_file=usuario.pd,
    output_df=sesionizar.pd, output_df1=enriquecer)
    - 1 Parsear(input_file=access.log, output_file=usuario.pd)

    - 1 Enriquecer(input_file=access.log, output_file=usuario.pd,
    output_df=sesionizar.pd, output_df1=enriquecer, output_df2=reporte.pd)

    - 1 Usuario(input_file=access.log, output_file=usuario.pd, output_df=sesionizar.pd)

This progress looks :) because there were no failed tasks or
missing external dependencies

===== Luigi Execution Summary =====
```

Una vez ejecutado, dentro de la ruta se crearan 5 archivos que son los outputs descritos con anterioridad *reporte.pd*, *sesionizar.pd*, *usuario.pd* y *enriquecer.csv*

Dentro de la página que abriste con anterioridad se tendrá la siguiente vista:





3.1 Ejecución Dashboard

Para poder ejecutar el dashboard es necesario que en la ruta especificada por el usuario (la cual debe ser la misma donde se obtuvieron los outputs y el archivo enriquecer.csv), es necesario tener la carpeta */R* la cual a su vez tiene dos archivos *server.R* y *ui.R*

Nuevamente en la línea de comandos, ejecutar:

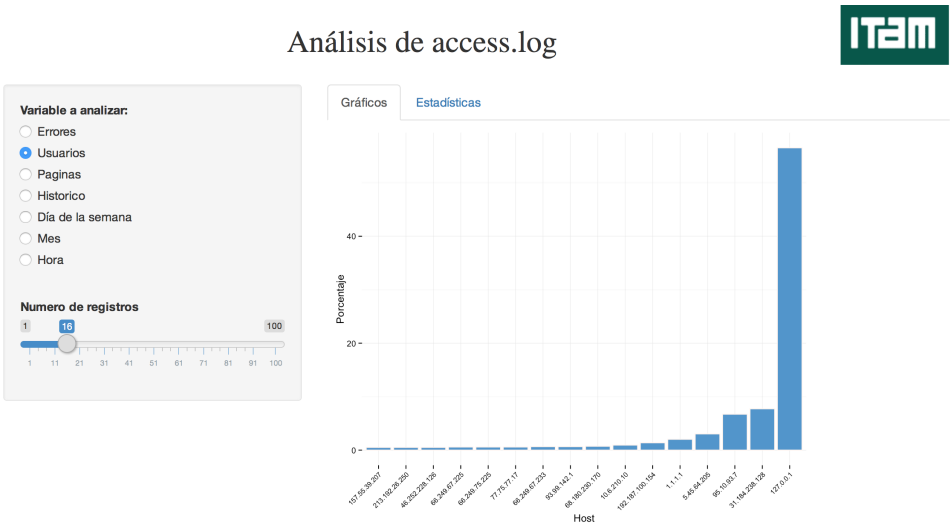
```
R -e "shiny::runApp('~/User/.../R/shinyapp')"
```

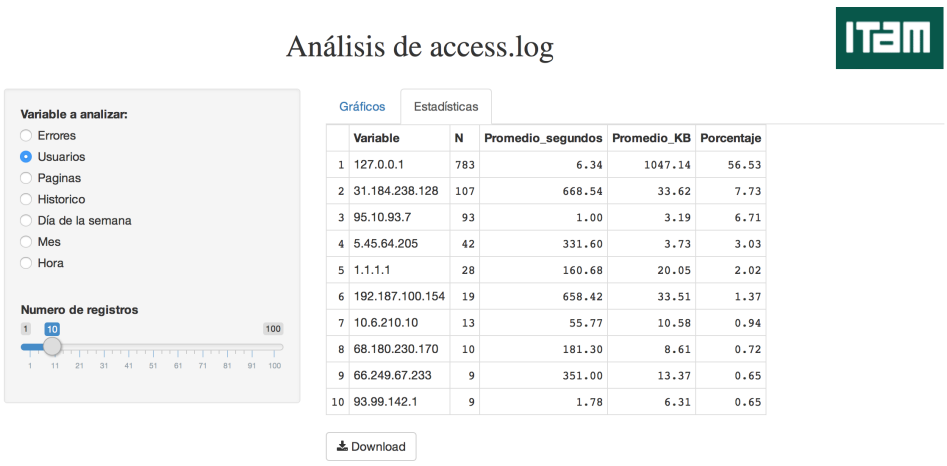
Al ejecutarse, se tendrá la siguiente salida:

```
> shiny::runApp('~/Desktop/conacyt/R/shinyapp')
Loading required package: shiny
This version of Shiny is designed to work with htmlwidgets >= 0.4.
Please upgrade your version of htmlwidgets.


Listening on http://127.0.0.1:5127
```

El puerto donde se ejecuta el dashboard será el hostname que se muestra, el cual se tendrá que copiar y pegar en el navegador para poder observar el dashboard.

Si todo es correcto, se mostrará lo siguiente:





En la parte del dashboard, además de poder visualizar las estadísticas de las variables más importantes, se podrá descargar en formato *csv* la consulta que le interese al usuario y si esta no le satisface, podrá descargar los insumos completos.

Referencias

Andersen, J., Larsen, R. S., Giversen, A., Pedersen, T. B., Jensen, A. H. y Skyt, J. (2000). Analyzing clickstreams using subsessions

Banerjee, A., y Ghosh, J. (2002). Characterizing visitors to a web site across multiple sessions. En Proceedings of NGDM'02: National Sci- ence Foundation Workshop on Next Generation Data Mining. Mar- riott Inner Harbor, Baltimore, MD, Estados Unidos.

Lagus, K. (2000). Text mining with the websom. Tesis doctoral, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finlandia.

http://httpd.apache.org/

https://wiki.duraspace.org/

http://www.edu4java.com/

http://www.programcreek.com/2013/04/what-is-servlet-container/