Instituto Tecnológico de Costa Rica

Escuela de Ingenieria en Computacion

Bases de datos ii

Grupo 40

Profesor Erick Hernandez

Estudiantes: Jose Pablo Murillo, Carlos Villalobos, Esteban Santamaría

Documentación Proyecto 2: Análisis de Tweets

I Semestre

2017

Introducción
El objetivo del proyecto es entender sobre el funcionamiento del filesystem de hadoop y el análisis de datos mediante map/reduce. Este proyecto consta de 3 partes. La primera parte consta de un programa en JAVA que recolecta, con la ayuda de la librería externa twitter4j, datos sobre tweets siendo publicados a tiempo real. Los datos se guardaran en un archivo de texto.

La segunda parte consta de un programa en Python que va a ser usado para correr el map reduce de Hadoop para analizar los datos y guardar estadísticas en una base de datos mySQL.

La tercera parte consta de una aplicación Web que colecta los datos guardados en la base de datos MySQL y genera gráficos de a partir de ellos.

Diseño y arquitectura

Aplicación JAVA

La aplicación JAVA se desarrolló usando Eclipse. Para la aplicación JAVA, se descargo la librería twitter4j que contiene lo necesario para poder recolectar información de tweets. Se usó la librería twitter4j-stream para que la aplicación consiguiera datos a tiempo real.

Para poder conectarse a Twitter, es importante tener una cuenta de twitter. A través de la cuenta de twitter se puede solicitar las “keys” que son usados para la configuración necesaria para ejecutar la aplicación.

El programa consta de dos clases. La primera clase es Main, que se ejecuta al inicio del programa. La segunda clase es la clase Listener que es la clase incluida en la librería Twitter4j que hay que implementar con sus respectivos métodos. La aplicación usa estas dos clases para recibir datos a tiempo real de los tweets y luego escribir información sobre ellos en el archivo “datos.txt”. Como se puede recibir datos con respecto a cualquier tema, para este proyecto se decidió usar un filtro para guardar información de tweets que tengan solo los siguientes hashtags:

- #2030NOW
- #women
- #costarica
- #puravida
- #MakeAmericaGreatAgain
- #TrumpRussia

- #RecycleReuse
- #TraficoCR

Diseño de la base de datos mySQL
Para este proyecto se diseñaron las siguientes tablas:

- tema(idTema, cantUsuarios, cantTweets): el idTema tiene una cantidad de usuarios cantUsuarios y una cantidad de tweets cantTweets.

- hashtags(hashtag,tema,apariciones): el hashtag no-tema aparece una cantidad de apariciones para un tema.

- incluirTema(idTema, temaIncluido, apariciones): El tema temaIncluido fue mencionado en el tema idTema y esto ocurrió en apariciones.

- palabra(palabra,tema,apariciones): La palabra fue mencionada en un tema y aparecio con un total de apariciones.

- tiempo(idTema,hora,apariciones): El tema idTema tuvo una cantidad de apariciones en cierta hora.

Aplicacion Python

Se uso Python para crear el archivo “mapperTweets.py” y “reducerTweets.py”. Ambas clases son usadas por hadoop para realizar el análisis de datos del archivo de texto y guardar las estadísticas en el motor de base de datos mySQL. Para la elaboración de la aplicación, se importaron los siguientes módulos:

- **sys:** de esta librería se usa sys.stdin para que el archivo reciba texto de la entrada estándar. Para hacer más fácil las validaciones necesarias de los datos, todos los tweets se convierten en su equivalente de mayúsculas.

- **ast:** de esta librería se usa ast.literal_eval para hacer la conversión de un string a una estructura o tipo de datos de Python equivalente.

- **mysql.connector:** Este módulo es usado para hacer la conexión con la base de datos mySQL y hacer las operaciones necesarias para guardar las estadísticas.

El archivo mapperTweets recibe texto por medio de la entrada estándar y luego imprime o “mapea” los datos requeridos que van a ser usados por el reducer para realizar los cálculos.

Para el mapper se definieron las siguientes funciones:

- sacarHora(tweet) : Recibe un tweet como texto y retorna la información con respecto a la hora en que se realizó el tweet.

- sacarUsuario(tweet): Recibe un tweet como texto y retorna el usuario que publicó el tweet.

- esHashtag(texto): Recibe un string y determina si es un hashtag de Twitter.

- esPalabra(texto): Recibe un string y determina si es una palabra válida.

- getHashtags(texto): Recibe un texto y retorna 3 listas: una sobre los temas encontrados en el tweet, otra sobre los otros hashtags, no temas, que se encontraron y la ultima la lista de palabras que se encontraron. El mapper no considera la preposiciones como palabras que debe mappear. Para esta validación se tomaron en cuenta una lista de preposiciones que se pueden ver en el archvo “ampperTweets.py”. Como hadoop transfiere la información por medio de key/value. El mapper usa la hora como key, y la lista de temas, hashtags, palabras y usuarios como value. Se decidió usar la hora porque es más fácil de ordenar que al haber escogido el tema u otro string.

El achivo ReducerTweets recibe texto y va guardando internamente estadísticas sobre los tweets en estructuras tipo Diccionario. En cuanto a las estadísticas se guarda lo siguiente:

- Número de usuarios que se recibieron por tema
- Número de tweets recibidos por tema
- Las palabras por tema y su total de apariencias por palabra.
- Los otros hashtags mencionados para cada tema con su total de apariencias por tema.
- Una distribución de tiempo donde se guarda cuántos tweets se hicieron por hora para cada tema.
- Para cada hashtag tema, cuantos tweets mencionan alguno de los otros temas.

Para guardar los datos se uso 1 estructura de Python que son las siguientes:

- Diccionario “dic” donde la llave es el tema y el valor es lo siguiente:
    - La cantidad de usuarios que mencionaron el tema
    - Una lista de cantidad de tweets realizados por hora
    - Una lista de palabras con su cantidad de apariencias

    - Una lista de otros hashtags con su cantidad de apariencias.

El archivo incluye las siguientes funciones:

- Analizar (secundarios): Toma los hashtags que no son temas y retorna una lista de los temas usando decode y code utf-8. Esto se usó debido a que en las pruebas se encontraban algunos errores ya que se encontraba imagenes, etc.

- getData(value): Recibe el valor pasado como texto por el mapper y retorna las listas “prinicipales”,que son los temas del tweet, “secundarios” que son los hasgtags no-temas, y palabras del tweet.

- ActualizarDiccionarioPrincipal: toma como parámetros la hora, el usuario, la lista de temas principales, la lista de hashtags no-temas y la lista de palabras. La función hace validaciones y actualiza los datos del diccionario “dic”.

- borrarBD: función que se encarga de limpiar los datos de las tablas en la base de datos mySQL.

- meterDatosTema: función que mete los datos de un tema y sus apariciones en la tabla “temas”.

- meterDatosPalabra: función que mete datos del top 10 de palabras con sus apariciones para un tema en la tabla “palabra”.

- meterDatosHashtags: función que mete datos del top 10 de hashtags para un tema en la tabla “hashtags”.

- meterDatosTiempo: función que mete la distribución por hora y sus apariciones correspondientes para un tema en la tabla “tiempo”.

- meterDatosIncluirPrincipales: función que mete los datos de la cantidad de veces que se mencionó un tema para un hashtag no-tema. Los datos se guardan en la tabla “incluirTema”.

Aplicación Web

La aplicación web consiste de 4 archivos:

- Html (Página a mostrar)
- Css (Modificación visual de la página)
- Javascript (Funcionalidades del sistema)
- PHP (Conecciones con la Base de Datos MySQL)

También se utilizan dos librerías externas para la visualización de gráficos:

- d3.v3.js
- c3.js

Funcionamiento:

Mediante Apache2 como servidor web, se montó un archivo php con contenido html para la estructura de la página.

Consistiendo de 6 botones independientes que habilitan cada función del sistema mediante un javascript y 8 opciones de checkbox como filtros de información.

Cada botón hace referencia a una función en el archivo javascript, mediante un jquery se le solicita al archivo php que se conecte a la base de datos MySQL y obtenga los datos nesesarios dependiendo de la operación. Estos datos son devueltos en un formato json para simplificar su análisis y acomodo.

La información es filtrada por medio de los checkbox y se solicita a la base únicamente la información correspondiente a los tags seleccionados por el usuario.

Una vez obtenida la información se crea una variable ‘chart’ la cual corresponderá al gráfico a mostrar.

Finalmente mostrado el gráfico al usuario, este podrá interactuar con él de manera dinámica.