Aplicación de Estructuras de Datos Probabilísticos en

Samuel Leiva[1], Junior Micha[2], Danilo Blas[3], Joel Janampa[4], Brener Bustillos[5]

***Resumen*—Este articulo matemático-computacional presenta los resultados obtenidos mediante un proceso de investigación sobre algunas estructuras de datos alternativas que pueden hacer que el trabajo de análisis de datos sea un más eficaz, antes de usar técnicas como un clúster de cómputo para ejecutar herramientas distribuidas de cómputo paralelo como por ejemplo Hadoop y Spark que son herramientas correctas, pero muy costosas.**

I. INTRODUCCIÓN

Este artículo describe tres estructura de datos probabilísticas: Bloom Fitler, Count-min sketch e HyperLogLog, programaremos cada uno en lenguaje R y/o Python y comprobaremos su funcionamiento para el conteo y las consultas referidas a elementos dentro de un conjunto.

I-A. Objetivos

- Aplicar la estructura bloom Fitler para hacer consultas sobre elementos que puedan pertenecer al conjunto analizado.
- Usar la estructura Count-min sketch.
- Calcular el número aproximado de elementos en un multiconjunto usando la estructura HyperLogLog.

En esta oportunidad usaremos cada una de las tres estructuras aplicadas sobre un problema real. En el primer (Bloom Filter) generaremos un conjuntos de n elementos aleatorios, seguidamente utilizaremos las funciones añadir y query(consultar) para añadir elementos y consultar si algún elemento pertenece al conjunto que hemos generado en el caso de la estructura Count-min sketch, finalmente con la estructura HyperLogLog generaremos un multiconjunto grande el cual calcularemos al cardinalidad de un elemento específico con una excatitud del 2 % usando 1,5kb de memoria.

Manuscrito creado el 21 de septiembre del 2018; cuya revisión final sera el 9 de Noviembre del 2018. Este trabajo es compatible en formato IEEE y se distribuye bajo el Proyecto LaTeX.El manuscrito puede ser encontrado en los github de los autores

[1] S. Leiva es estudiante de pregrado de ciencias de la computación, Universidad Nacional de Ingeniería, 2015-2021, Lima, Peru. https://github.com/SamuelLeiva

[2] J. Micha es estudiante de pregrado de matemática, Universidad Nacional de Ingeniería, 2016-2022, Lima, Perú. https://github.com/JMicha23

[3] D. Blas es estudiante de pregrado ciencias de la computación ,Universidad Nacional de Ingeniería, 2015-2021, Lima,Perú. https://github.com/Sdann26

[4] J. Janampa es estudiante de pregrado de matemática, Universidad Nacional de Ingeniería, 2015-2021, Lima, Perú. https://github.com/JoelJanampaBautista

[5] B. Bustillos es estudiante de pregrado de matemática, Universidad Nacional de Ingeniería, 2015-2021, Lima, Perú. https://github.com/brenner-08

I-B. BLOOM FITLER

- Estructura de datos espacio eficiente (Borton Howard Bloom -1970).
- Usado para probar si un elemento es miembro de un conjunto.
- Una consulta arroja “posiblemente en el conjunto”(falso positivo) o “definitivamente no en el conjunto”(falso negativo).
- Se puede agregar elementos al conjunto pero no removerlos.
- Mayores elementos con añadidos , mayores probabilidad de falsos positivos .
- Generalmente , menos de 10 bits por elemento son requeridos para un 1 % de probabilidad falso positivo , independientemente del tamaño o número de elementos del set .

1. **Descripción de un algoritmo**: Arreglo de m bits , todo en 0 . También debe haber k funciones hash , cada uno de los cuales mapea algún elemento a una de las m posiciones (distribución aleatoria uniforme).
2. **Para añadir un elemento**: Se alimenta a cada una de las k funciones para obtener k posiciones en el arreglo . Todas esas posiciones se van a 1 .
3. **Para consultar(si es que el elemento está)**: Se alimenta a cada una de las k funciones para obtener k posiciones en el arreglo , si algunas de las posiciones son 0 el elemento definitivamente no se encuentra en el conjunto , si todos son 1, entonces o bien el elemento está o los bits han sidos puestos a 1 cuando fueron insertados.

I-C. COUNT-MIN SKETCH

- Sirve como una tabla de frecuencia de eventos en una data stream . Usa funciones hash para mapear eventos a frecuencias , pero solo usa espacios sublineales a costa de ”sobrecontar algunos eventos a causa de colisiones (2003-Graham Cormode , S.Muthu Muthokrishnan).
- Esencialmente es lo mismo que Bloom fitler, pero son usados de manera diferente y se ponen el tamaño de manera diferente.

1. **Estructura**: El objetivo de count-min sketch es la de consumir un stream de eventos , uno a la vez , y contar la frecuencia de los diferentes tipos de eventos en el stream . En cualquier momento el sketch puede ser consultado para la frecuencia de un particular tipo de evento y regresará un estimado de esta frecuencia dentro de una cierta distancia de la frecuencia real , con una cierta probabilidad.

I-D. HYPERLOGLOG

- Algoritmo para el conteo distintivo , aproximando el número de distintos elementos en un multiconjunto . Es capaz de estimar cardinalidades mayor a 10^9 con una certeza del 2 % usando 1,5 kb de memoria.

1. **Algoritmo**: Se basa en la observación de que las cardinalidades de un multiconjunto de números aleatorios uniformemente distribuidos pueden ser calculados estimando el máximo número de ceros en la representación binaria de cada número en el conjunto . Si el máximo número de ceros observados es n , un estimado para el número de elementos en el conjunto es 2^n .
2. **Operaciones**
   Add: Añade un elenento al conjunto.
   Count: Nos arroja la cardinalidad el conjunto.
   Merge: Para obtener la unión de dos conjuntos.

II. ESTADO DEL ARTE

- **"Theory and Practice of Bloom Filters for Distributed Systems"**
  Este articulo presenta una serie de técnicas probabilísticas como los Bloom-filters y sus variantes como stable Bloom Filter, Adaptative Bloom Filters, Filter Banks, etc. que se utilizan para reducir el procedimiento de la información y los costos de esta información.

- **.An Improved Data Stream Summary: The Count-Min Sketch and its Applications"**
  Este articulo presenta otras aplicaciones de la estructura de datos Count-min Sketch para problemas como encontrar cuartiles, elementos frecuentes, etc.

- **"HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm"**
  En este articulo se presentan mejoras al algoritmo HyperLogLog reduciendo sus requisitos de memoria y aumentar su precisión para un rango importante de cardinalidades.

III. DISEÑO DEL EXPERIMENTO

III-A. Implementación del Bloom Filter

1. Crear una estructura de datos Bloom Filter.
2. Generar un arreglo de elementos con la operación Add.
3. Realizar consultas(operación Query) para saber si un elemento en particular se encuentra en nuestro arreglo.

III-B. Implementación del Count-min Sketch

1. Crear una estructura de datos Count-min Sketch.
2. Generar un arreglo bidimensional con eventos como elementos.
3.

III-C. Implementación del HyperLogLog

1. Crear una estructura de datos Hyperloglog
2. Generación de un multiconjunto(arreglo) aleatorio con números como elementos.
3. Calculamos la cardinalidad del conjunto de números creados.

III-D. Ecuaciones

sdajflas ecuanciones

$$\alpha + \beta = \chi \tag{1}$$

Note that the equation is centered using a

III-E. subitituo

-
-
-

IV. INTRODUCCION

RELLENAR

IV-A. PIRMER

IV-B. LLENAR SI SE DESEA

wiiiiisdshdfksbsdka

Figura 1. descripcion de la imagen

V. CONCLUSIONS

APPENDIX

rellenear

ACKNOWLEDGMENT

REFERENCIAS

[1]
[2]
[3]