Proyecto Bloom Filter

Estudiante: Braulio César Partida Meneses
Profesor: Fernando Esponda Darlington

Introducción

Los bloom filters son estructuras de datos que se utilizan para almacenar conjuntos de datos de forma eficiente. Son muy útiles para aplicaciones donde es importante poder determinar si un elemento está presente en el conjunto, pero no es necesario saber si está presente o no.

El tamaño de un bloom filter depende de la cantidad de elementos que se almacenarán en él y del porcentaje de falsos positivos que se estén dispuestos a aceptar. En este proyecto, se desarrolló un método para calcular el tamaño mínimo de un bloom filter dada la cantidad de elementos y un porcentaje de falsos positivos.

Además, se realizó un experimento para ver qué le pasaba al bloom filter si se aumentaba el número de hash funciones (k).

Hipotesis

Al aumentar el número de hash funciones (k), el porcentaje de falsos positivos disminuye, pero el tamaño del bloom filter aumenta.

Materiales

1. Base de datos que continente urls maliciosas o benignas
2. IDE en este caso vscode y RStudio
3. Estructura de datos BloomFilter

Procedimiento

Funciones:

La siguiente función calcula la tasa de falsos positivos para un bloom filter de un tamaño dado:

```
def calcFalsoPosi(estan, noEstan, n: int, k: int, cantNoEstan: int) ->
float:
    bloom = BloomFilter(n, k)
    focount = 0
    for idx, elem in estan:
        bloom.inserta(elem)
    for idx, elem in noEstan:
        if bloom.busca(elem):
            focount += 1
    return focount / cantNoEstan
```

La función calcFalsoPosi toma los siguientes parámetros como entrada:

- **estan**: Un conjunto de elementos que se sabe que están en el bloom filter.
- **noEstan**: Un conjunto de elementos que se sabe que no están en el bloom filter.
- **n**: El tamaño del bloom filter.
- **k**: El número de hash funciones que se utilizarán en el bloom filter.
- **cantNoEstan**: La cantidad de elementos que no están en el bloom filter.

La función devuelve la tasa de falsos positivos.

La siguiente función calcula el tamaño mínimo de un bloom filter para un porcentaje de falsos positivos máximo dado, usa la lógica de busqueda binaria:

```
def calcMinM(fPMAX: float, estan, noEstan, n: int, k: int) -> str:
    nMin = 1
    nMax = n * 3
    cantNoEstan = noEstan.size
    p = 0
    while True:
        nMid = (nMin + nMax) // 2
        enumEstan = enumerate(estan["url"])
        enumNoTan = enumerate(noEstan["url"])
        fP = calcFalsoPosi(enumEstan, enumNoTan, nMid, k, cantNoEstan)
        print(f"m:{nMid}, falsoPositivo:{fP}")

        if fP <= fPMAX and fP >= fPMAX - 0.0001:
            return f"{nMid}, {fP}"
        elif p == nMid:
            return f"{nMid}, {fP}"

        if fP > fPMAX:
            nMin = nMid
        else:
            nMax = nMid

        p = nMid
```

La función calcMinM toma los siguientes parámetros como entrada:

- **fPMAX**: El porcentaje de falsos positivos máximo permitido.
- **estan**: Un conjunto de elementos que se sabe que están en el bloom filter.
- **noEstan**: Un conjunto de elementos que se sabe que no están en el bloom filter.
- **n**: El número de elementos que se almacenarán en el bloom filter.
- **k**: El número de hash funciones que se utilizarán en el bloom filter.

Devuelve el tamaño mínimo del bloom filter y la tasa de falsos positivos

Experimentación:

El código siguiente le un conjunto de datos de un archivo CSV llamado data.csv. El conjunto de datos contiene información sobre sitios web, incluidos sus tipos. Los sitios web se clasifican como "benignos" o "maliciosos".

El código luego divide el conjunto de datos en dos conjuntos:

- **datos_insertar**: Un conjunto de sitios web maliciosos.
- **datos_revisar**: Un conjunto de sitios web benignos.

Los conjuntos de datos se guardan en archivos CSV separados.

El código luego itera sobre un rango de valores de k, que es el número de hash funciones que se utilizarán en el bloom filter. Para cada valor de k, el código llama a la función **calcMinM**() para calcular el tamaño mínimo del bloom filter necesario para lograr un porcentaje de falsos positivos del 1%.

El código guarda los resultados de cada iteración en un diccionario llamado d. El diccionario contiene las siguientes claves:

- **k**: El valor de k utilizado para la iteración.
- **m**: El tamaño mínimo del bloom filter calculado para la iteración.
- **fP**: El porcentaje de falsos positivos calculado para la iteración.

El código luego crea un DataFrame a partir del diccionario d. El DataFrame se guarda en un archivo CSV llamado resultados.csv.

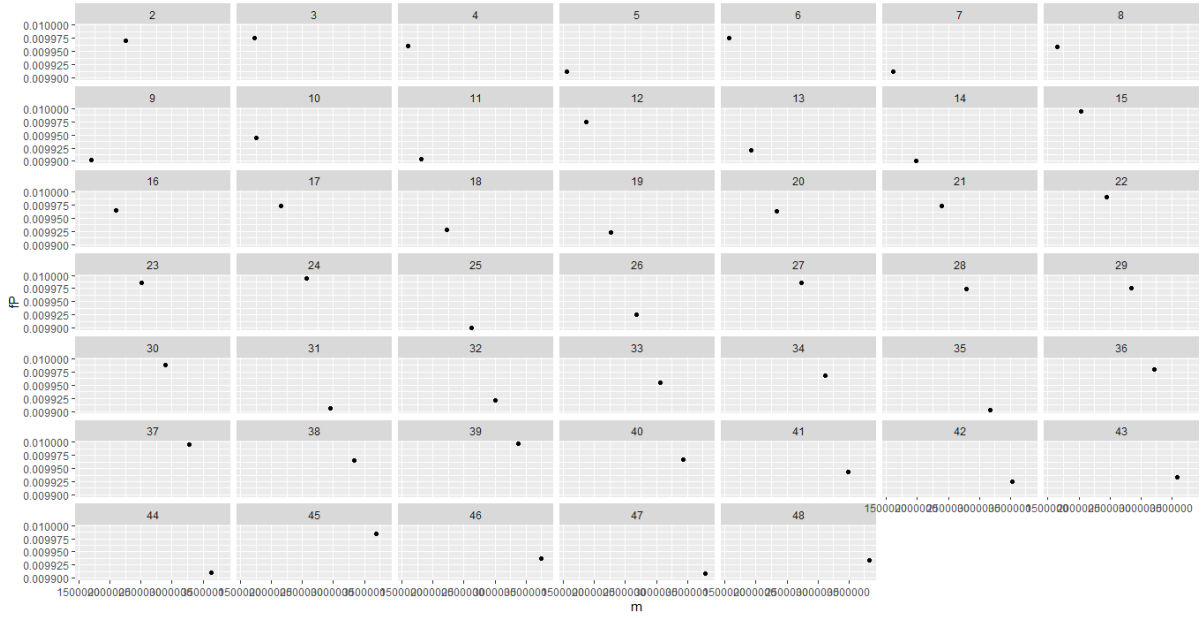
El codigo:

```
df = pd.read_csv("ProyectoBloom/CSVs/data.csv")

df["malicious"] = df["type"].apply(lambda x: 1 if x != "benign" else 0)


datos_insertar = df[df["malicious"] == 1]
datos_revisar = df[df["malicious"] == 0]
datos_insertar.to_csv("ProyectoBloom/CSVs/insercionBF.csv",
index=False)
datos_revisar.to_csv("ProyectoBloom/CSVs/revisarBF.csv", index=False)


d = {"k": [], "m": [], "fP": []}

for k in range(1, 60):
    resp = calcMinM(0.01, datos_insertar, datos_revisar,
datos_revisar.size, k).split(", ")
    d["k"].append(k)
    d["m"].append(resp[0])
    d["fP"].append(resp[1])

    print("\n")

datosResultantes = pd.DataFrame(d)
```

```
datosResultantes.to_csv("ProyectoBloom/CSVs/resultados.csv",
index=False)
```

Despues sobre el documento tipo csv resultante se uso R para graficar y conseguir un mejor análisis, usando la librería ggplot2

Análisis de resultados



La tabla anterior muestra la relación entre la cantidad de hash (k), el tamaño del bloom (m) y los falsos positivos (fp) en un filtro de Bloom.

En general, podemos observar que:

- La cantidad de hash (k) tiene un efecto directo en el tamaño del bloom (m). A medida que aumenta k, también aumenta m. Esto se debe a que el filtro de Bloom necesita más bits para representar más funciones hash.
- La cantidad de hash (k) tiene un efecto indirecto en los falsos positivos (fp). A medida que aumenta k, disminuye la probabilidad de falsos positivos. Esto se debe a que el filtro de Bloom es más preciso cuando se usa un mayor número de funciones hash.

Gracias a la función summary() en R se puede conseguir la siguiente informacion

```
       k               m                fP        
 Min.   : 2.0   Min.   :1557726   Min.   :0.009900  
 1st Qu.:13.5   1st Qu.:2011122   1st Qu.:0.009922  
 Median :25.0   Median :2623028   Median :0.009961  
 Mean   :25.0   Mean   :2635182   Mean   :0.009951  
 3rd Qu.:36.5   3rd Qu.:3249026   3rd Qu.:0.009975  
 Max.   :48.0   Max.   :3830351   Max.   :0.009997  
```

En particular, podemos observar que:

- El valor mínimo de k es 2. Esto significa que el filtro de Bloom solo usa dos funciones hash. En este caso, el tamaño del bloom es de 1557726 bits y la tasa de falsos positivos es de 0,99%.
- El valor máximo de k es 48. Esto significa que el filtro de Bloom usa 48 funciones hash. En este caso, el tamaño del bloom es de 3830351 bits y la tasa de falsos positivos es de 0,999%.

Conclusión

Por lo tanto, podemos concluir que, para reducir la tasa de falsos positivos en un filtro de Bloom, es necesario aumentar la cantidad de hash. Sin embargo, esto también aumenta el tamaño del bloom, lo que puede tener un impacto en el rendimiento del filtro.