CS 5968/6968: Data Str Alg Scalable Comp	Spring 2023
Lecture 9 — Feb 22, 2023	
Prof. Prashant Pandey	*Scribe: Shreyas Singh*

1 Overview

In the last lecture, we had a guest lecture on Locality Sensitive Hashing.

In this lecture, we are going to cover Filters.

2 Introduction

- A filter is an approximate representation of a set that trades off accuracy (for existence) for better space requirements
- It **guarantees a bounded false +ve rate** of ϵ where $0 < \epsilon < 1$
- There are no false negative queries (1-sided errors)
- They are deterministic with the results of queries

$$\begin{aligned} &\text{if } q \in S \text{ return ``Exists'' with } P = 1 \\ &\text{if } q \notin S \text{ return ``Exists'' with } P \leq \epsilon \text{ (false +ve)} \\ &\qquad\qquad \text{return ``Does not Exist'' with } P > 1 - \epsilon \end{aligned}$$

Types: Static Filters: Cannot insert new items after the creation of filter (XOR Filters, Ribbon Filter)
Dynamic Filters: Can be viewed as a streaming solution (Bloom Filter, Quotient Filter, Cuckoo Filter)

Space: Dictionaries (hash tables) take $\boldsymbol{\Omega(n \log |U|)}$ bits
Filters take $\geq \boldsymbol{n \log \frac{1}{\epsilon}}$ as a lower bound (does not depend on $\boldsymbol{|U|}$)
Generally, $\epsilon = 2\%$ so a filter requires ≈ 8 bits/item
Therefore, filters are very useful in cases where $|U - S|$ is very large.

3 Use Cases

- Filters can be used to represent an approximate key set of databases that can fit in RAM and avoid negative queries from generating expensive disk accesses. For these scenarios they have to be able to support inserts, queries and deletes.
- A 6 TB database of 512 B keys ($\approx$ 12.88 Billion Keys) can be reduced to a 12 GB filter using 6 37 bit has functions ($\approx$ 1 B/Key) with a false positive rate of 1.56%

4 de Bruijn Graph

Nodes k-length strings

Edge Join two nodes with a $k-1$ overlap
where $k-1$ suffix of node 1 $= k-1$ prefix of node 2
Value of the edge is going to be a $k+1$ length string which is a combination of the two nodes it joins

Genomics Each node is going to be a k-mer with edges connecting 1-base extensions
Each node has an out-degree of 4 as each extension can be one of four base pairs
Using a hash table to store the nodes as keys and edges as values would require a large amount of space
By just storing the edges ($k+1$ length strings) is an extremely efficient way of storing a de Bruijn graph of k-mers [1]
Query: Search for $k-1$ length suffix and prefix
Traverse: Need number of hops based on queries
False +ve will give an edge that does not exist (causing topological errors, see Figure 4) - but can be removed using a weighted de Bruijn graph along with using graph algorithms (like Max-Flow/Min-Cut)

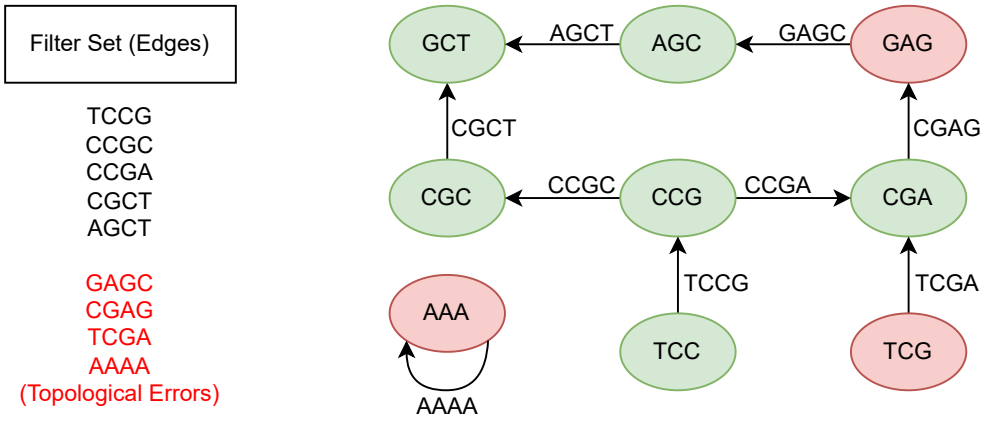

Figure 1: Examples of Topological Errors introduced due to the false positives present in a filter. The false positives in the filter set (red edge names) introduce extra nodes that are not present in the original graph (red nodes)

5 Bloom Filter

- Bit Vector of size m and k hash functions as a filter implementation
- Space $\approx 1.44n\log(\frac{1}{\epsilon})$ bits
- Bloom filters do no support deletes as doing so as is would affect other values and possibly introduce a false -ve

- Counting Bloom filters would be able to support deletes but are not very space efficient [2]
- False +ve analysis:

$$\begin{aligned}
\Pr(\text{a certain bit is not set 1 by a certain hash function}) &= 1 - \frac{1}{m} \\
\Pr(\text{a certain bit is not set 1 by any hash function}) &= \left(1 - \frac{1}{m}\right)^k \\
\Pr(\text{a certain bit is not set 1 after n insertions}) &= \left(1 - \frac{1}{m}\right)^{kn} \\
\boxed{\lim_{m\to\infty}\left(1 - \frac{1}{m}\right)^m = \frac{1}{e}} & \\
&= e^{\frac{-kn}{m}} \\
\Pr(\text{a certain bit is 1 after n insertions}) &= 1 - e^{\frac{-kn}{m}} \\
\Pr(\text{all k bits are 1 during a query of item}) &= \left(1 - e^{\frac{-kn}{m}}\right)^k \\
&\approx \text{False Positive Rate}
\end{aligned}$$

- Using more space reduces false +ve rate while adding more items increases false +ve rate.
- For a given value of m, n

$$\boxed{k = \frac{m}{n}\ln 2}$$

$$\boxed{\epsilon = 2^{-k}}$$

References

[1] Pandey, Prashant, et al. "deBGR: an efficient and near-exact representation of the weighted de Bruijn graph." Bioinformatics 33.14 (2017): i133-i141.

[2] Pandey, Prashant, et al. "A general-purpose counting filter: Making every bit count." Proceedings of the 2017 ACM international conference on Management of Data. 2017.