Bloom Filters

CS 277
Brad Solomon

February 22, 2023
Lab_quacks Feedback

Average score: 75%, 85%

PL average time: 90 minutes

Genuinely didn’t think it would be a hard assignment.

You must be familiar with stacks and queues for the next exam.
Learning Objectives

Build a conceptual understanding of a bloom filter

Review probabilistic data structures and one-sided error

Formalize the math behind the bloom filter

Introduce extensions to the bloom filter
Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects in a memory-constrained environment?

Constrained by Big Data (Large $N$)

<table>
<thead>
<tr>
<th>Sky Survey Projects</th>
<th>Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPOSS (The Palomar Digital Sky Survey)</td>
<td>3 TB</td>
</tr>
<tr>
<td>2MASS (The Two Micron All-Sky Survey)</td>
<td>10 TB</td>
</tr>
<tr>
<td>GBT (Green Bank Telescope)</td>
<td>20 PB</td>
</tr>
<tr>
<td>GALEX (The Galaxy Evolution Explorer)</td>
<td>30 TB</td>
</tr>
<tr>
<td>SDSS (The Sloan Digital Sky Survey)</td>
<td>40 TB</td>
</tr>
<tr>
<td>SkyMapper Southern Sky Survey</td>
<td>500 TB</td>
</tr>
<tr>
<td>PanSTARRS (The Panoramic Survey Telescope and Rapid Response System)</td>
<td>~ 40 PB expected</td>
</tr>
<tr>
<td>LSST (The Large Synoptic Survey Telescope)</td>
<td>~ 200 PB expected</td>
</tr>
<tr>
<td>SKA (The Square Kilometer Array)</td>
<td>~ 4.6 EB expected</td>
</tr>
</tbody>
</table>

Table: http://doi.org/10.5334/dsj-2015-011

Estimated total volume of one array: 4.6 EB
Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects in a memory-constrained environment?

Constrained by resource limitations

- cache: < 1 second
- RAM: Hours - Days
- disk: Months
- network: Years

(Estimates are Time x 1 billion courtesy of https://gist.github.com/hellerbarde/2843375)
Reducing storage costs

1) Throw out information that isn’t needed

2) Compress the dataset
Reducing a hash table

What can we remove from a hash table?

$H(k_1) = i_1$
Reducing a hash table

What can we remove from a hash table?

Take away values
Reducing a hash table

What can we remove from a hash table?

Take away values and keys
Reducing a hash table

What can we remove from a hash table?

Take away values and keys

This is a **bloom filter**
Bloom Filter: Insertion

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]

\[ h(k) = k \% 7 \]
Bloom Filter: Insertion

An item is inserted into a bloom filter by hashing and then setting the hash-valued bit to 1

If the bit was already one, it stays 1
Bloom Filter: Deletion

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]

\[ h(k) = k \% 7 \]

\_delete(13)

\_delete(29)
Bloom Filter: Deletion

Due to hash collisions and lack of information, items cannot be deleted!
Bloom Filter: Search

S = {16, 8, 4, 13, 29, 11, 22}

h(k) = k % 7

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

_find(16)
_find(20)
_find(3)
Bloom Filter: Search

The bloom filter is a *probabilistic* data structure!

If the value in the BF is 0:

If the value in the BF is 1:
Probabilistic Accuracy: Malicious Websites

Imagine we have a detection oracle that identifies if a site is malicious.

- **Not malicious**
- **Malicious**
Probabilistic Accuracy: Malicious Websites

Imagine we have a detection oracle that identifies if a site is malicious

True Positive:

False Positive:

False Negative:

True Negative:
Imagine we have a **bloom filter** that **stores malicious sites**…

<table>
<thead>
<tr>
<th></th>
<th>Bit Value = 1</th>
<th></th>
<th>Bit Value = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item Inserted</strong></td>
<td></td>
<td><strong>Item NOT inserted</strong></td>
<td></td>
</tr>
<tr>
<td><strong>True Positive</strong></td>
<td>$H(z)$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>False Negative</strong></td>
<td>0</td>
<td>1</td>
<td>‘Yes’</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Probabilistic Accuracy: One-sided error

We will NEVER have a False Negative:

We will get some False Positives:

Dataset:

Query:

search with one-sided error
Probabilistic Accuracy: One-sided error

Query:

Dataset:

search with one-sided error

Query:

search with one-sided error

…
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter

\[ h_1 \quad h_2 \quad h_3 \quad \ldots \quad h_k \]
Bloom Filter: Repeated Trials

\[ h_{\{1,2,3,\ldots,k\}}(y) \]
Bloom Filter: Repeated Trials

If any query yields 0, item is not in the set

$h_{\{1,2,3,\ldots,k\}}(y)$
Bloom Filter: Repeated Trials

If all queries yield 1, item may be in the set; or we might have collided \( k \) times.
Bloom Filter: Repeated Trials

Using repeated trials, even a very bad filter can still have a very low FPR!

If we have $k$ bloom filter, each with a FPR $p$, what is the likelihood that all filters return the value ‘1’ for an item we didn’t insert?
Bloom Filter: Repeated Trials

But doesn’t this hurt our storage costs by storing $k$ separate filters?

$h_1$  $h_2$  $h_3$  ...  $h_k$
Bloom Filter: Repeated Trials

Rather than use a new filter for each hash, one filter can use $k$ hashes

$S = \{ 6, 8, 4 \}$

$h_1(x) = x \% 10$ \hspace{1cm} $h_2(x) = 2x \% 10$ \hspace{1cm} $h_3(x) = (5+3x) \% 10$
Bloom Filter: Repeated Trials

Rather than use a new filter for each hash, one filter can use \( k \) hashes

\[
\begin{align*}
h_1(x) &= x \mod 10 \\
h_2(x) &= 2x \mod 10 \\
h_3(x) &= (5 + 3x) \mod 10
\end{align*}
\]

\_find(1)

\_find(16)
Bloom Filter

A probabilistic data structure storing a set of values

Built from a bit vector of length $m$ and $k$ hash functions

Insert / Find runs in: _______________

Delete is not possible (yet)!
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function

What is our expected FPR after $n$ objects are inserted?
Bloom Filter: Error Rate

Given bit vector of size $m$ and 1 SUHA hash function

What's the probability a specific bucket is 1 after one object is inserted?

Same probability given $k$ SUHA hash function?
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function $h\{1,2,3,...,k\}$

Probability a specific bucket is 0 after one object is inserted?

After $n$ objects are inserted?
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function

What's the probability a specific bucket is 1 after $n$ objects are inserted?
Bloom Filter: Error Rate

Given bit vector of size \( m \) and \( k \) SUHA hash function

What is our expected FPR after \( n \) objects are inserted?

The probability my bit is 1 after \( n \) objects inserted

\[
\left( 1 - \left( 1 - \frac{1}{m} \right)^{nk} \right)^k
\]

The number of [assumed independent] trials
Bloom Filter: Error Rate

Vector of size $m$, $k$ SUHA hash function, and $n$ objects

To minimize the FPR, do we prefer...

(A) large $k$ 

(B) small $k$

\[
\left( 1 - \left( 1 - \frac{1}{m} \right)^{nk} \right)^k
\]
Bloom Filter: Optimal Error Rate

**Claim:** The optimal hash function is when \( k^* = \ln 2 \cdot \frac{m}{n} \)
Bloom Filter: Optimal Error Rate

Claim: The optimal hash function is when $k^* = \ln 2 \cdot \frac{m}{n}$

\[
\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k \approx \left(1 - e^{-\frac{nk}{m}}\right)^k
\]

\[
\frac{d}{dk} \left(1 - e^{-\frac{nk}{m}}\right)^k \approx \frac{d}{dk} \left(k \ln(1 - e^{-\frac{nk}{m}})\right)
\]

Derivative is zero when $k^* = \ln 2 \cdot \frac{m}{n}$
Bloom Filter: Error Rate

\[(1 - e^{-nk/m})^k\]

\[k^* = \ln 2 \cdot 10 = 6.93\]
Bloom Filter: Optimal Parameters

\[ k^* = \ln 2 \cdot \frac{m}{n} \]

Given any two values, we can optimize the third

\[ n = 100 \text{ items} \quad k = 3 \text{ hashes} \quad m = \]

\[ m = 100 \text{ bits} \quad n = 20 \text{ items} \quad k = \]

\[ m = 100 \text{ bits} \quad k = 2 \text{ items} \quad n = \]
Bloom Filter: Optimal Parameters

\[ m = \frac{nk}{\ln 2} \approx 1.44 \cdot nk \]

Optimal hash function is still \( O(n) \)!

\( n = 250,000 \) files vs 260 TB

\( n = 60 \) billion — 130 trillion
Bloom Filter: Website Caching

Loaded this before?

0
1
0
1
0
1

Cache this page!

Add to filter (but don’t cache!)

Sequence Bloom Trees

Imagine we have a large collection of text…

And our goal is to search these files for a query of interest…
Bloom Filters: Unioning

Bloom filters can be trivially merged using bit-wise union.

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 0 \\
4 & 0 & 4 & 0 \\
5 & 0 & 5 & 0 \\
6 & 1 & 6 & 1 \\
7 & 0 & 7 & 1 \\
8 & 0 & 8 & 1 \\
9 & 1 & 9 & 1 \\
\end{array}
\]

\[ U = \]

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 0 \\
4 & 0 & 4 & 0 \\
5 & 0 & 5 & 0 \\
6 & 1 & 6 & 1 \\
7 & 0 & 7 & 1 \\
8 & 0 & 8 & 1 \\
9 & 1 & 9 & 1 \\
\end{array}
\]
Sequence Bloom Trees

SRA 00001  SRA 00002  SRA 00003  SRA 00004  SRA 00005  SRA 00006  SRA 00007  SRA 00008
Sequence Bloom Trees

Are $\geq \theta$ fraction of query kmers $\in$ this Bloom filter?

If YES, move to children

If NO, stop looking at this subtree (Global mismatch)
SBT per-query time was recorded using a maximum of a single filter in each subtree, but estimated accuracy on a set of 100 random files ranged from 95–100% (Supplementary Table 4). Three collections of representative bacteria metagenomic runs were used to measure the performance of SBTs with varying sensitivity threshold values.

Measuring the performance of SBTs can speed up the use of algorithms, such as STAR or SRA-BLAST, for fast search of large transcriptomic databases. A single bloom tree search took 3.3 d using a single thread on the NIH cluster (Supplementary Fig. 7). This time was reduced to 5 min with 16 CPU cores (Supplementary Fig. 8).

**DISCUSSION**

Currently, it is difficult to access all the relevant data relating to a particular strain of bacteria. Fast search of this type will be essential to make good use of the ever-growing collection of available sequencing data. Researchers could search for conditions that can be revealed only through the analysis of multiple data sets from individual laboratories and sequencing centers to support large-scale sequence searches, not just for RNA-seq data, but for genomic and metagenomic collections as well.

Research groups are collecting data at a rapid pace, and face the same difficulties as the public genome database as more researchers are generating data. Mining of these data and could be used to uncover biological insights that can be revealed only through the analysis of multiple data sets from individual laboratories and sequencing centers to support large-scale sequence searches, not just for RNA-seq data, but for genomic and metagenomic collections as well.

Both false positives and false negatives can arise from a mismatch between SBT’s definition of present (coverage of the query) and Sailfish’s definition of expressed (as estimated fraction of the query). Both definitions are related, but not perfectly aligned, resulting in some disagreement that is quantified by the false-positive rates (FPR) and false-negative rates (FNR).

The median true-positive rate across all queries was 100% for all comparisons, and the median false-positive rate was 0.2% for queries that return at least one file. The median FNR for queries that return at least one file was 96–100% across the three data sets. The operational characteristics of SBT are summarized in Supplementary Table 4.

The SBTs enable the efficient mining of these data and could be used to uncover biological insights that can be revealed only through the analysis of multiple data sets from individual laboratories and sequencing centers to support large-scale sequence searches, not just for RNA-seq data, but for genomic and metagenomic collections as well.


Bloom Filters: Tip of the Iceberg


There are many more than shown here…