Assignment 4: Frequent Items

Christopher Mertin/u1010077
cmertin@cs.utah.edu
March 13, 2017

Overview

In this assignment you will explore finding frequent items in data sets, with emphasis on techniques designed to work at enormous scale.

You will use two data sets for this assignment:

- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A4/S1.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A4/S2.txt`

These data sets each describe a set of $m = 1{,}000{,}000$ characters. The order of the file represents the order of the stream.

As usual, it is highly recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory: `http://www.cs.utah.edu/~jeffp/teaching/latex/`

1 Streaming Algorithms

A (20 points): Run the Misra-Gries Algorithm (see **L11.3.1**) with $(k-1) = 9$ counters on streams S1 and S2. Report the output of the counters at the end of the stream.

In each stream, from just the counters, report how many objects *might* occur more than 20% of the time, and which must occur more than 20% of the time.

	a	b	c	o	p	v	f	r	l	h	j	w
`S1.txt`	$195,715$	$155,715$	$105,715$	2	1	1	1					
`S2.txt`	$245,715$	$135,715$	$175,715$					1	1	1	1	1

We can find the upper bound of the count by

$$C_j + \frac{(m/k)}{2}$$

with C_j being the j^{th} counter, m being the stream size, and k being the total number of counters. We can solve for this to see which ones are over a certain percentage, meaning $0.20 \times 1,000,000 = 200,000$, for seeing whcih *might be* in 20% of the file. $\frac{(m/k)}{2} = 55,556$, when plugging in our numbers. In checking the above counts, a and b are *maybe* up to 20% of `S1.txt`, while a and c are *maybe* up to 20% of `S2.txt`.

B (20 points): Build a Count-Min Sketch (see **L12.1.1**) with $k = 10$ counters using $t = 5$ hash functions. Run it on streams S1 and S2.

For both streams, report the estimated counts for objects a, b, and c. Just from the output of the sketch, which of these objects, with probably $1 - \delta = 31/32$, *might* occur more than 20% of the time?

	a	b	c
`S1.txt`	$283,300$	$259,251$	$176,539$
`S2.txt`	$303,588$	$220,401$	$233,553$

To determine which have a 20% chance with probability $1 - \delta = 31/32$, we use the expression for the Markov Inequality, given as

$$\hat{f}_j - m/k \leq \hat{f}_j \leq f_a \leq \hat{f}_a + m/k$$

where $m/k = 100,000$. As we are using 5 hash functions, the probability of $1 - \delta = 31/32$ is already satisfied since $2^5 = 32$. In bounding the above values by the above equation we get

	a	b	c
`S1.txt`	$183,300 \leq f_a \leq 383,300$	$159,251 \leq f_b \leq 359,251$	$76,539 \leq f_c \leq 276,539$
`S2.txt`	$203,588 \leq f_a \leq 403,588$	$120,401 \leq f_b \leq 320,401$	$133,553 \leq f_c \leq 333,553$

To be 20%, it has to have a count of $200,000$, so we can say the following from the above table:

- `S1.txt`: a and b are likely to be 20% while c is unlikely
- `S2.txt`: a is likely to be 20% while b and c might be, though c is more likely than b

C (5 points): How would your implementation of these algorithms need to change (to answer the same questions) if each object of the stream was a "word" seen on Twitter, and the stream contained all tweets concatenated together?

- We could implement a k-gram on the input while streaming. Then the group of k characters can be treated as elements being counted.

D (5 points): Describe one advantage of the Count-Min Sketch over the Misra-Gries Algorithm.

- As hashed values of the smaller/less frequent values will be added and subtracted, you can still get a rough idea of values outside of k.

2 BONUS

The exact heavy-hitter problem is as follows: return *all* objects that occur more than 10% of the time. It cannot return any false positives or any false negatives. In the streaming setting, this requires $\Omega(\min\{m, n\})$ space if there are n objects that can occur and the stream is of length m.

A: (1 point) A 2-Pass streaming algorithm is one that is able to read all of the data in-order exactly twice, but still only has limited memory. Describe a small space algorithm to solve the exact heavy hitter problem, i.e., with $\varepsilon = 0$, (say for $\phi = 10\%$ threshold).

- We could optimize the number of counters by using the first pass to count the number of unique items in the stream and then create that many labels. Then, when doing the second pass using one of the above streaming algorithms to count the objects with the exact number of counters/labels if using Misra-Gries.