Comp 480/580: Assignment #2

Rice University — Due Date: Monday, 10/21/2025

1 Count-Min Sketches, Count-Median Sketches, and Count-Sketches

Task: We will implement Count-Min Sketches, Count-Median Sketches (replace minimum with median in count-min sketches), and Count-Sketches. The plan is to compare their performance on the heavy hitter problem. We will use the query field in AOL dataset **(url)** to generate a stream of words and the task is to find most frequent words in this search log using the sketches and also compare them with the exact counts.

Implementation: We have to implement four things 1.) Count-Min Sketch. 2) Count-Median Sketch, 3) Count-Sketch and 4) A plain dictionary to keep track of exact counts for evaluation.

1.1 Guidelines

Processing:

- Assignment 1 will come handy; we will use the same AOL dataset. So your codes and hash functions written in Assignment 1 will be directly useful.
- Download Aol dataset, which includes anonymized user ids and clicks data. **(url)**
- Data prepossessing: This time, we will only use query keywords fraom this file. So if the keywords are "pop up ads," then we treat it as three words, and we will insert all the three words into our sketch, with count 1 each.
- For all the keywords, we will insert it into our sketch and dictionary one by one.

Sketches: For sketches, we will implement insert and query functionality, which will insert a token in the sketch, which is simply adding "1" to the count, every-time a token is observed (Streaming model). The query functionality will return an estimate of the frequency (or total count) of a given token. The sketches as two parameters, 1.) d, which is the number of the hash functions, and 2) R, which is the range of the hash functions. For this assignment, we will keep d to be fixed at value 5. We will experiment with the three values of R $\{2^{10},\ 2^{14},\ 2^{18}\}$

Dictionary: Keep a standard dictionary of tokens and count. Implement it in any convenient way you like. (this might be heavy on memory, so be careful. There are around 400,000 unique query in the dataset.)

Frequent, Random, and Infrequent Tokens: Using the dictionary, find 100 most frequent tokens, call it *Freq-100* along with their counts. Similarly, find 100 most infrequent tokens (*Infreq-100*) and get their counts. Also, get 100 random tokens from the dictionary *Rand-100* along with their counts. It is OK if *Rand-100* has any overlap with *Freq-100* or *Infreq-100*. Note the space used by the dictionary.

Plots the errors: Now for every value of range R, we will generate three plots of errors, one each for *Freq-100*, *Rand-100* and *Infreq-100*. The x-axis is the 100 words (sorted based on their frequency. The y-axis is the error defined as $\frac{abs(\hat{C}_i - C_i)}{C_i}$, where $\hat{C}_i$ is the estimated count from the sketch and C_i is the true count from the dictionary. In each of the plot, we will have three lines, one for the Count-Min Sketch, one for count-median sketch, and finally the other for the Count-Sketch. Thus in total, we have 3 x 3 = 9 plots.

Sketches with Heaps: Finally, add the functionality of returning the identity of approximate top-500 items from your sketches. We will maintain a little dictionary of top-500 along with their counts in a min-heap. For each of the sketches (count-min and count-sketch), while updating the count of a token, estimate the token's total count so far and check it with the smallest value in the little dictionary. If the token is already among the top 500, update the count. If not, then check with the minimum count and update if found better. After all the insertions, report the top-500 elements. Plot the size of the intersection of top-500 from the sketch with the actual top-100 from the dictionary (Yes, we report 500 approximate frequent and see how many of 100 most frequent did we identify). The plot will have the x-axis as the values of R and y-axis as the size of intersection. We need one plot with three lines, one for count-min, one of count-median and last one for count-sketch.

1.2 What to Submit:

In about a page report, describe your findings and write conclusions. Please add plots to the report and make observations on every plot you present.

Submit codes, plots, report, makefile (if any), and the main script to run the code that generates the plots. Submit one compressed file via Canvas. We will only type the texts in the command line, and it should generate all the outputs. Make sure you fix the random seeds so that multiple runs of the code produces the same output.