ENS de Lyon
Computer Science Department

Internship Report

Efficient sketching for frequency estimation for heavy-tail distributions

Name: Collas Estéban
Program: L3 Computer Science
Academic Year: 2024–2025
University Supervisor: François Schwarzentruber
Lab Tutor: Gregory Kucherov

Host Laboratory: Gaspard Monge Computer Science Laboratory (LIGM)
Internship Period: From June 2 to July 28, 2025

Done in Carrières-sous-Poissy, on December 2, 2025

Symbol	Description
S	Stream (ie. an ordered list)
$S[j]$	j^{th} element of S
$\widetilde{S}$	Set of all the elements in S
$\lvert \cdot \rvert$	Cardinality of a set or a stream
N	$\lvert S \rvert$ (except in the Hot and Cold case, where $\lvert S \rvert \sim \mathcal{P}(N)$)
E	Number of distinct elements from which we generate the stream
m	Number of counters per row
k	Number of rows, also number of hash functions
x_k	k-th most frequent element in S
η_x	Number of occurrences of x in S
$\widehat{\eta}_x$	Estimation on η_x
η_x^j	Number of occurrences of x in $S[:j]$ (ie. in the j first elements of S).
$\widehat{\eta}_x^j$	Estimation on η_x^j
h_i	Hash function of the i^{th} row
$h_i(x)$	Address of the cell where x is hashed in row i
$\widetilde{h_i}(x)$	Value of the cell $h_i(x)$
s_i	Sign function of the i^{th} row
$\varepsilon_i(x)$	Error of x in row i
α	Zipf's proportionality constant
$\mathcal{H}$	Set of hot elements
$\mathcal{C}$	Set of cold elements
p_h	Probability of drawing an arbitrary hot element
p_c	Probability of drawing an arbitrary cold element
C_h	Probability of drawing a given hot element
C_c	Probability of drawing a given cold element
μ_h	Expected number of occurrences of a given hot element
μ_c	Expected number of occurrences of a given cold element
λ_h	Hot density
λ_c	Cold density
σ_h^2	Variance of the Gaussian approximation of the error if $x \in \mathcal{H}$
σ_c^2	Variance of the Gaussian approximation of the error if $x \in \mathcal{C}$

Table 1: Summary of the notations used in the entire rapport.

1 Introduction

In the age of big data, applications across diverse fields such as network routing, large-scale databases, and bioinformatics are confronted with the challenge of processing massive data streams. These streams, which can be seen as unbounded sequences of data, often arrive at high speeds, making it infeasible to store them in their entirety. A fundamental task in stream analysis is frequency estimation: determining the number of occurrences of each distinct element. The naive approach of using a hash map or dictionary is often impractical due to prohibitive memory requirements, as it would need to store a unique identifier and a counter for every distinct element observed.

To overcome these memory limitations, the field of data sketching provides powerful probabilistic data structures. These algorithms maintain a compact summary, or "sketch," of the data, often in sub-linear space, from which properties of the original stream can be estimated with high accuracy. Foundational algorithms such as the Count-Min Sketch and the Count-Sketch are widely used for frequency estimation. They offer formal guarantees on their estimation error,

trading a small degree of uncertainty for significant gains in memory efficiency.

However, the classic analyses of these algorithms typically provide worst-case error bounds that hold for any arbitrary data stream. They often do not leverage any prior knowledge about the statistical properties of the data being processed. This raises a critical question: if we have information about the underlying generative distribution of the stream—for instance, knowing that it follows a power law like the Zipf distribution, or a simpler bimodal distribution—can we design more precise estimators that outperform the standard ones? This report investigates this very problem, postulating that incorporating distributional knowledge into the estimation process can significantly reduce error.

This report makes several contributions to this question. First, we conduct a detailed analysis of the Count-Sketch algorithm within the context of a "Hot & Cold" generative model, a simplified framework for heavy-tailed distributions. Based on this analysis, we derive novel estimators for item frequencies using Maximum Likelihood (MLE) and Maximum A Posteriori (MAP) principles. We then present experimental results that compare the performance of these new estimators against the standard median-based estimator. Finally, we explore the theoretical possibility of extending this analytical framework to the more complex and realistic Zipf distribution, identifying the conditions under which such an analysis is tractable.

This report is organized as follows. Section 2 provides the necessary background on classic sketching algorithms and statistical principles. Section 3 reviews the state of the art in the analysis of these structures. Section 4 presents our main results, including the experimental comparison and the detailed analysis of the Hot & Cold model. Finally, Section 5 concludes by summarizing our findings, discussing limitations, and suggesting avenues for future research.

2 Preliminaries

2.1 Classic Algorithms

The best-known data sketching algorithms are Count-Min Sketch, Count-Sketch, Bloom Filter, and Hyperloglog. Except for the latter, those algorithm are quite similar. Let us (almost in publication order) introduce them, and some more.

In those algorithm, I will talk about the "point query". That is "what we seek when we ask information about an element to the sketch". Also, the hash functions we use in the following text need to have good properties, like pairwise independance, in order to distribute the elements from the stream uniformly into the sketch.

2.1.1 Bloom Filter

The simplest of them is probably the Bloom Filter (BF, Bloom (1970)). (see figure 1).

Let S be a stream, t a moment in time. We denote by $S(t)$ the set of elements seen so far. The BF models $S(t)$ - "point query" on the BF returns whether an element is in $S(t)$. Referring back to the introduction, one could do this with a dictionary where values are always 1. This would have a space complexity of $O(|S(t)| \cdot \psi)$, where ψ is the size taken by an element of S. We face the two problems mentioned earlier.

A solution would be to create a fixed-size dictionary — a pair (T, h) where T is an boolean array of size m (initially $[false] * m$) and h is a hash function. For each element x in S, we set $T[h(x)]$ to $true$. To check if an element x is in $S(t)$, we look at $T[h(x)]$. This is a data sketching algorithm with spatial complexity $O(m)$.

The problem is that we need to choose $m >> |S(t)|$ to hope for accurate modeling of $S(t)$.

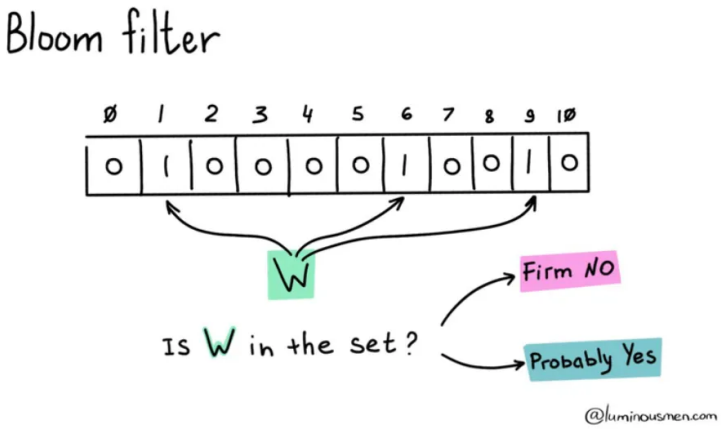

Figure 1: Bloom Filter. Source: luminousmen.com

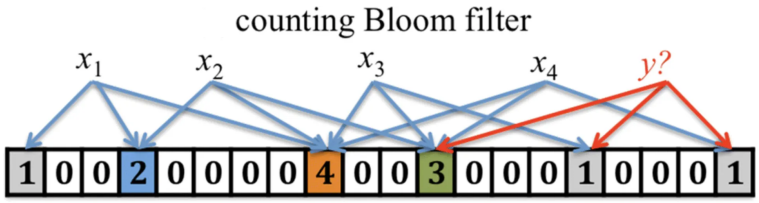

Figure 2: Counting Bloom Filter. Source: Medium

The BF improves this approach. We define a $(k+1)$-tuple $(T, h_1, ..., h_k)$, where T is an array of size m (initially $[false] * m$), and $h_1, ..., h_k$ are hash functions. Each element of $S(t)$ is hashed into T (that is we set $h_1(x), ..., h_k(x)$ to $true$). To check if an element x is in $S(t)$, we test if all the values at the addresses $h_i(x)$ $(=: \widetilde{h_i}(x))$ are $true$.

It can be shown that $m = O(|S(t)|)$ suffices to ensure a good success probability (it is for example in Cohen and Matias (2003)).

We must also store the hash functions in $O(k)$. In real cases, this takes little space.

An interesting property of the Bloom Filter is that it can only produce false positives, never false negatives. This is particularly valuable in real-world applications, such as distributed databases. It is used in Google Bigtable, PostgreSQL and for Bitcoin, for example.

2.1.2 Counting Bloom Filter

One drawback of BF is that we cannot remove elements from the constructed set. If we add and then remove an element x, resetting all Booleans hashed by x to False may inadvertently remove another element y.

The Counting Bloom Filter (CBF, Fan et al. (2000)) is a variant that allows this functionality (see figure 2). It updates a counter array instead of Booleans. When adding/removing an element from the stream, we increment/decrement all hashed positions. A "point query" returns true iff all counters are non-zero (we do nothing when removing an element not in the set, so counters never go negative).

2.1.3 Spectral Bloom Filter

The Spectral Bloom Filter (SBF, Cohen and Matias (2003)) is an extension of the CBF (it is still the structure we see in figure 2). It stems from the observation that the CBF actually

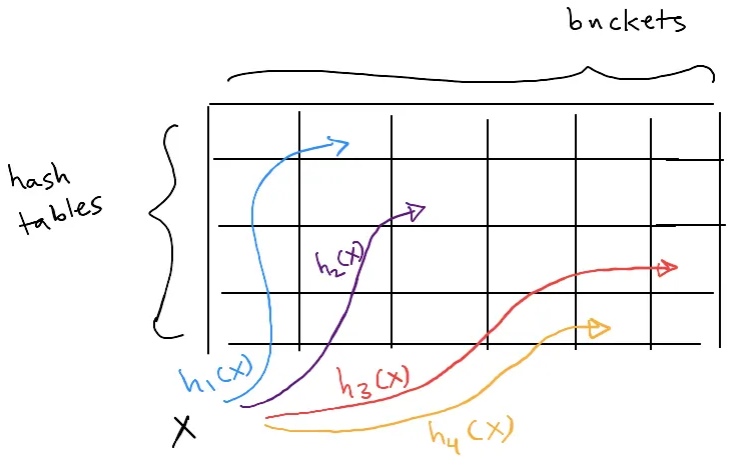

Figure 3: Count Min Sketch. Source: Medium

sketches a multiset. In the SBF, we no longer remove elements. We add elements as before, and a "point query" asks for the number of occurrences of an element in the stream, approximated by $\min\{\widetilde{h}_i(x)\}_{i=1}^{k}$.

2.1.4 Count Min Sketch

The Count Min Sketch (CM, Cormode and Muthukrishnan (2005a)) is very close to the SBF (see figure 3. The only difference is that, unlike previous sketches, it uses a matrix with k rows — one array per hash function. Each function hashes elements into its respective array. This is equivalent to partitioning the original array into k disjoint parts and restricting the image of the i^{th} hash function to the i^{th} part. CM also sketches a multiset, and a "point query" returns $\min\{\widetilde{h}_i(x)\}_{i=1}^{k}$.

In both CM and SBF, the estimated number of occurrences is always greater than or equal to the real number. This mirrors the BF's property of estimating a superset.

There is a variant, the Conservative Count Min (CMS, Cormode and Muthukrishnan (2005b)), which works the same way, except we only increment the counters that are minimal. This greatly reduces error (see Fusy and Kucherov (2022), Mazziane et al. (2022) for more details). This concept firstly appeared in the article presenting the Spectral Bloom Filter (Cormode and Muthukrishnan, 2005b).

2.1.5 Count Sketch

In real-world applications, it may be desirable to always overestimate and never underestimate, as in CM. However, this leads to biased estimations, meaning the expected estimate of the number of occurrences of an element x differs from its actual count (specifically, it is strictly greater).

To obtain an unbiased heuristic, one can use Count Sketch (CS, Charikar et al. (2002)). It works like CM, but in addition to hash functions $h_1, ..., h_k$, we define sign functions $s_1, ..., s_k$, each associated with a row of the sketch matrix (see figure 4. The s_i functions map the sketch elements to -1 or $+1$. When adding an element x to the sketch, instead of adding 1 to the address cells $(1, h_1(x)), (2, h_2(x)), ..., (k, h_k(x))$, we add $s_i(x)$ to the cell at address $(i, h_i(x))$.

The "point query" on the CS returns $median\{s_i(x) \cdot \widetilde{h}_i(x)\}_{i=1}^{k}$, and is used to estimate the number of occurrences of the element x.

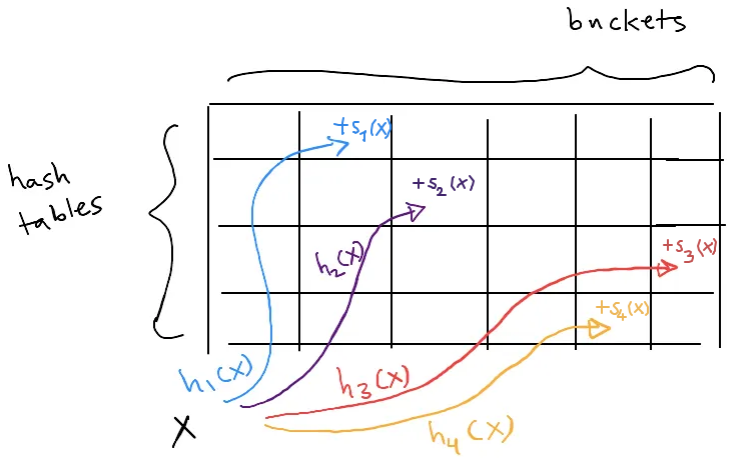

Figure 4: Count Sketch. Source: Medium (edited by me)

We indeed get an unbiased heuristic:

Let η_x be the number of occurences of x in S.

With S, x, i fixed and s_i, h_i random:

$$\mathbb{E}\left(s_i(x) \cdot \widetilde{h}_i(x)\right) = \mathbb{E}\left(\eta_x + \sum_{x' \in S \setminus \{x\}} \eta_{x'} \cdot \mathbb{1}_{\{h_i(x)=h_i(x')\}} \cdot s_i(x) s_i(x')\right) \tag{1}$$

$$= \mathbb{E}(\eta_x) + \sum_{x' \in S \setminus \{x\}} \eta_{x'} \cdot \frac{1}{m} \cdot 0 \tag{2}$$

$$= \eta_x \tag{3}$$

My internship was not really about the Flajolet Martin and Hyperloglog Algorithms, but I introduced them in the appendix to show the variety there is in data sketching algorithms.

2.2 Zipf Law

The Zipf law, very close to the power law and the Pareto law, is a classic law used to test sketching and big data algorithms. It models many observable phenomena, such as the frequency of words in natural language text.

Let $(z, E) \in (\mathbb{R}^+ \times \mathbb{E})$. Zipf(z,E) is a distribution over $[|1, E|]$, such that if $X \sim$ Zipf(z,E): $\forall k \in [|1, E|], P(X = k) \propto \frac{1}{k^z}$ (the proportionality factor is $\alpha := \frac{1}{\sum_k \frac{1}{k^z}}$).

(see figures 5 and 6).

We can also take $E = +\infty$, but then we must have $z > 1$ for the series to converge.

Intuitively, X is likely to be small (the probability decreases as $\frac{1}{k^z}$):

This means that on a log-log scale, we will get a straight line. One can also notice that $\frac{k}{k+1} \underset{k \to \infty}{\to} 1$. Thus, high-rank elements all have similar frequencies, whereas low-rank elements have very different frequencies from each other.

To test an algorithm that takes a stream as input, we define the stream $S = (X_i)_i$, where $\forall i, X_i \sim Zipf(z, E)$, for z and E chosen depending on the context, and run our algorithm on the stream S. We do this in the section 4.1.

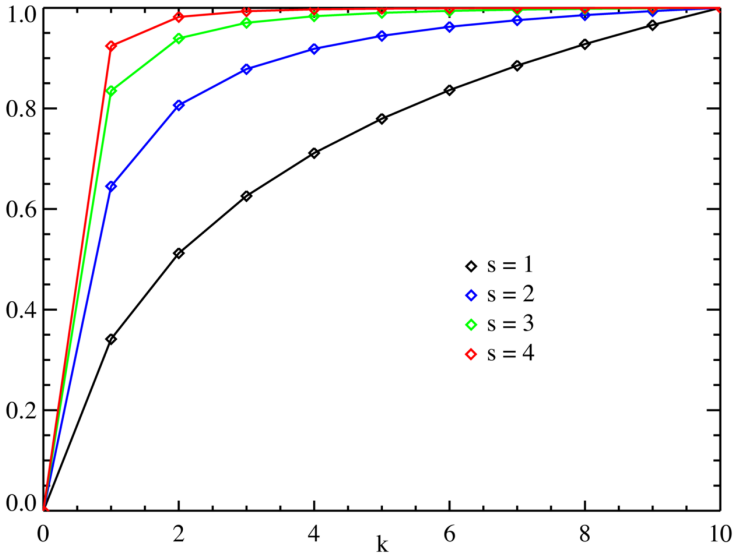

Figure 5: Zipf(s,10), cumulative distribution function. Source: Wikipedia

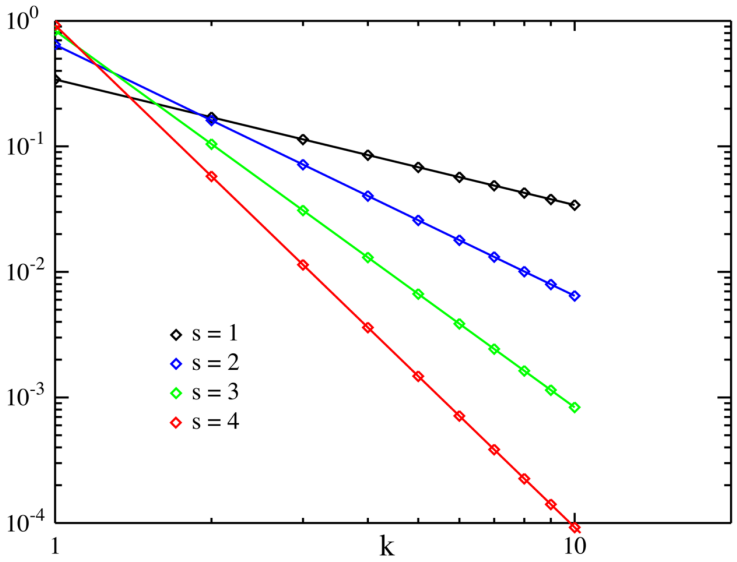

Figure 6: Zipf(s,10), probability mass function, log-log scale. Source: Wikipedia

The strong connections between Pareto, power laws, and the Zipf law are clearly developed in Adamic (2000).

2.3 Some Principles of Bayesian Probability

Bayesian probability relies on the Bayes formula, which allows one to update a prior belief $P(A)$ in light of some information B:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{4}$$

$$= \frac{P(B \mid A)\, P(A)}{\sum_C P(B \mid C)\, P(C)} \tag{5}$$

The second equality assumes that the events C form a partition of the space of possibilities (in

the discrete case).

2.4 Maximum A Posteriori Estimation (MAP)

The Maximum A Posteriori (MAP) estimator seeks to determine the value of the parameter θ that maximizes the posterior probability $P(\theta \mid D)$, where D denotes the observed data.

Discrete Case

If θ follows a discrete probability distribution, then the posterior is given by Bayes' formula:

$$P(\theta \mid D) = \frac{P(D \mid \theta)\, P(\theta)}{P(D)} \tag{6}$$

and the MAP estimator is written as:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} P(\theta \mid D) \tag{7}$$

$$= \arg\max_{\theta} \left[\frac{P(D \mid \theta)\, P(\theta)}{P(D)} \right] \tag{8}$$

$$= \arg\max_{\theta} \left[P(D \mid \theta)\, P(\theta) \right] \tag{9}$$

(because $P(D)$ does not depend on θ).

Continuous Case

If θ is a real continuous parameter, we work with probability densities. Let $f(D \mid \theta)$ be the likelihood density of the data given θ, and $\pi(\theta)$ the prior density of the parameter. The posterior density is then proportional to:

$$\pi(\theta \mid D) \propto f(D \mid \theta)\, \pi(\theta) \tag{10}$$

and the MAP estimator is:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} f(D \mid \theta)\, \pi(\theta) \tag{11}$$

2.5 Maximum Likelihood Estimator

Using a prior is not always convenient, so the Maximum Likelihood Estimator uses a uniform prior and returns:

$$\arg\max_{\theta} f(D \mid \theta) \tag{12}$$

This is less precise if we had reasons to use a non-uniform prior, but it can be simpler to use, and feels more natural in some cases.

3 State of the Art

3.1 Original Articles

3.1.1 Original Analysis of Count Sketch

Originally, Count Sketch was created to solve the FindCandidateTop problem (Charikar et al., 2002):

Algorithm 1: FindCandidateTop(S, k, ℓ)

Input: A stream S, two integers k and ℓ

Output: A list of ℓ elements from S such that the k most frequent elements in S are included in the list

However, the authors note that this problem is very hard to solve. Indeed, let x_i be the i-th most frequent element in S. If we imagine that $\eta_{x_k} = \eta_{x_{l+1}} + 1$, i.e., the k-th most frequent element has almost the same frequency as the $(l+1)$-th, it will be nearly impossible to find only l elements that are likely to include the k most frequent ones.

Thus, the authors introduce FindApproxTop:

Algorithm 2: FindApproxTop(S, k, ε)

Input: A stream S, an integer k, and a positive real number ε

Output: A list of k elements from S such that each element x in the list has a number of occurrences $\eta_x > (1-\varepsilon)\eta_{x_k}$

In this article, Charikar et al. show that Count Sketch can solve FindApproxTop with space complexity $O\left(k \log\left(\frac{|S|}{\delta}\right) + \frac{\sum_{q' \in I} \eta_{q'}^2}{(\varepsilon \eta_{x_k})^2} \log\left(\frac{|S|}{\delta}\right)\right)$,

where

- k is the number of elements we want to find
- δ is the desired accuracy (the algorithm returns a suitable set with probability at least $1-\delta$)
- ε controls the accuracy of the returned set
- I is the set of all the elements in S that are not in the top k of the most frequent ones.

The article also shows that with the same complexity, we have the following result:

Theorem 1. *With probability at least $1-\delta$, for all $j \leq |S|$, we have:*

$$\left|\text{median}\{\widetilde{h}_i(x) \cdot s_i(x)\} - \eta^j_{S[j]}\right| \leq 8\gamma$$

where $\eta^j_{S[j]}$ is the number of occurences of $S[j]$ in $S[:j]$ (the j first elements of S), $\gamma := \sqrt{\frac{\sum_{q' \in I}^{m} n_{q'}^2}{b}}$ and I is the set of all the elements in S that are not in the top k of the most frequent ones.

3.1.2 Original Analysis of Count Min

In the Count Min presentation paper, Cormode et al. showed the following theorem (Cormode and Muthukrishnan, 2005a) :

Theorem 2. *Taking $k := \lceil \ln(\frac{1}{\delta}) \rceil$ hash functions, and $m := \lceil \frac{e}{\varepsilon} \rceil$ counters per hash function, the estimator $\hat{\eta}_x$ (the minimum of the counters) satisfies the following properties:*

$$\eta_x \leq \hat{\eta}_x \quad \text{and, with probability at least } 1-\delta, \quad \hat{\eta}_x \leq \eta_x + \varepsilon F_1.$$

where $F_1 := \sum_{x \in \widetilde{S}} \eta_x$

Using this theorem, we thus have a spatial complexity for storing the sketch in $O(dw) = O\left(\ln\left(\frac{1}{\delta}\right)\frac{e}{\varepsilon}\right)$, to which must be added the storage of the hash functions, in $O(d)$.

3.2 Comparison of Count Sketch and Count Min

Actually, a different analysis (Scribe and Kent) leads to the following table:

	Space	**Error**
Count-Min	$O\left(\frac{1}{\varepsilon}\log m\right)$	εF_1 (one-sided bound)
Count-Sketch	$O\left(\frac{1}{\varepsilon^2}\log m\right)$	$\varepsilon\sqrt{F_2}$ (two-sided bound)

where

$$\begin{cases} \varepsilon \text{ is the tolerated error} \\ F_1 := \sum_{x \in \widetilde{S}} \eta_x \\ F_2 := \sum_{x \in \widetilde{S}} \eta_x^2 \end{cases}$$

That is, Count Sketch has a slightly higher spatial complexity, but generally better accuracy than Count Min.

3.3 More Precise Proofs

There is way more articles about Count Min than Count Sketch, mainly because both data structure are very similar and Count Min is often the easier of the two to analyze.

Count Sketch and Count Min often perform better than the bounds found in the original articles describing these algorithms. This can be seen in the graphs from (Chen et al., 2021).

In fact, there have been analysis to improve the bounds on Count Min and Count Sketch (Minton and Price, 2014).

3.4 Variants and Extensions

There are two important parts in a data sketching algorithm: (i) the sketch in itself, that is what we keep in memory and (ii) the function we apply to get information about the underlying data.

We saw ways to change (i) earlier. But it is also possible to change part (ii) of the sketching algorithm.

For example, for the Count Min Sketch, Ting (2018) and Chen et al. (2021) both explored how we can use a different estimator than the minimum to estimate the real occurences. The approach of both papers is to estimate the law of the error in a given address in the sketch. That is, the sum of the increments which comes from collisions, and not from the element we are interested in. Ting's paper use the addresses where our element is not hashed to to estimate it's value, whereas Chen's paper estimates this error using a more ad-hoc algorithm.

4 Results:

(All the results presented here are reproductible, the seed used is in the github repo.)

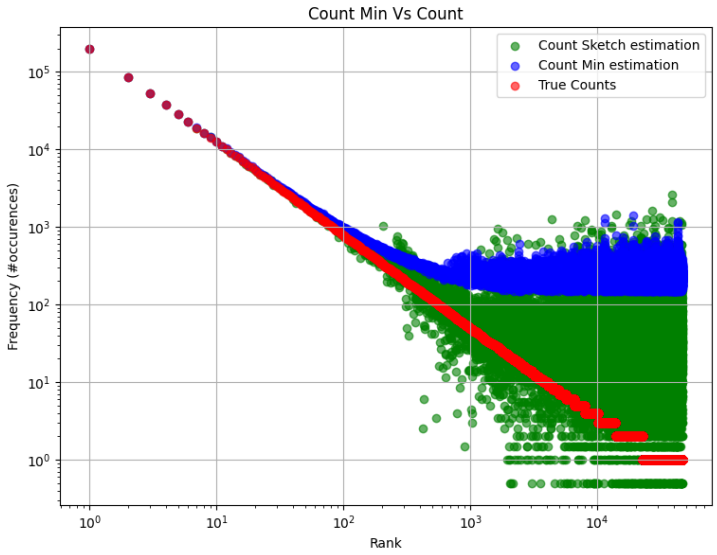

Figure 7: Count Min Vs Count Sketch. Zipf(10^5, **1.2**), $N = 10^6$, $k = 5$, $m = 800$, **non-conservative** Count Min. Log-log scale.

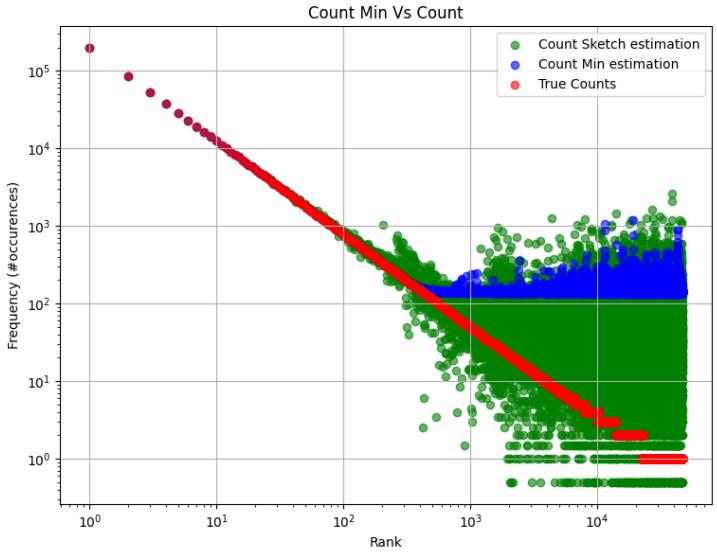

Figure 8: Count Min Vs Count Sketch. Zipf(10^5, **1.2**), $N = 10^6$, $k = 5$, $m = 800$, **conservative** Count Min. Log-log scale.

4.1 Comparison Between Count Sketch & Count Min:

My internship started with getting hands-on experience with Count Sketch and Count Min data structures, implementing them in Python (here: https://github.com/Ae-rys/code_data_sketching_algorithms) and generating plots showing the behavior of the two data structures on a stream generated according to a Zipf distribution.

(see figures 7 and 8)

In red, one recognizes the shape of a Zipf law. In the Count Min sketches, one observes in both graphs what is known as the "waterfall effect." There is a plateau. Indeed, starting from a certain rank (around ~ 500 here), all the counters are equal. This is a consequence of the law of large numbers and the Zipf distribution.

When looking at the Count Sketch, it seems that more and more values appear above the

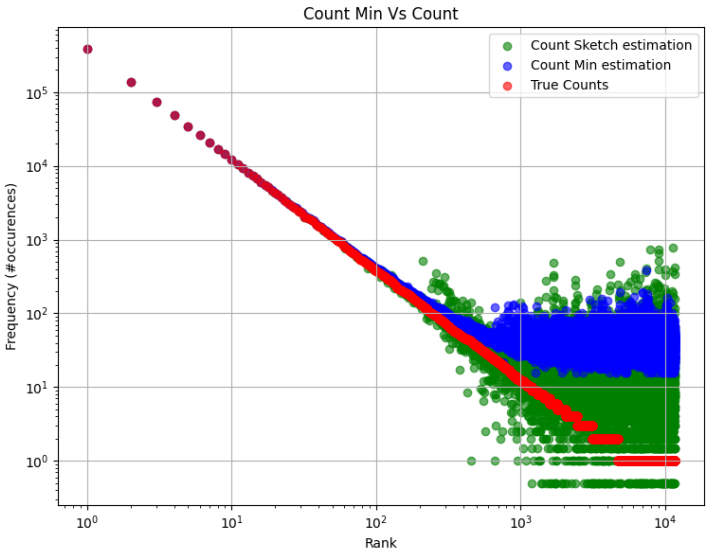

Figure 9: Count Min Vs Count Sketch. Zipf(10^5, **1.5**), $N = 10^6$, $k = 5$, $m = 800$, **non-conservative** Count Min. Log-log scale.

"waterfall" as the rank increases. In reality, this is most likely an illusion due to the log-log scale. Indeed, points get closer together as the rank increases, so the points above the "waterfall" appear less spaced out than they actually are (a similar phenomenon occurs for the Count Min Sketch).

It is also noticeable that something happens around rank 500 in all three models (around the "waterfall"), which seems to indicate a real phenomenon in the frequencies starting from this rank. My hypothesis is that this comes from the earlier remark about the Zipf law, which stated that $\frac{k}{k+1} \to_{k\to\infty} 1$. I believe that in the case $z = 1.2$, from around rank ~ 500, the elements have frequencies that are very difficult to distinguish.

To support this hypothesis, one can observe that the breaking point shifts to the right when z increases (since in this case, the ratio $\left(\frac{k}{k+1}\right)^z$ between the frequencies of ranks k and $k+1$ converges more slowly to 1) (see figures 9).

There is more data on the comparison between CM and CS in (Cormode and Muthukrishnan, 2005b).

I also played a little bit with Count Min and Spectral Bloom Filter and showed they had similar capabilities.

4.2 MLE with Monte Carlo

For this algorithm, I was exploring the behaviour of Count Min under a stream generated using a Zipf law and, especially, whether knowing the stream followed a Zipf law could help us recover the real occurences with more precision.

Let $\widehat{\eta}_x := \min\{\widetilde{h}_i(x)\}_{i=1}^k$.

I was inspired by D. Ting's article (Ting, 2018) to create a variant of CM. Indeed, D. Ting estimates "on-the-fly" the distribution of the error on a cell of the sketch, and then uses an MLE to estimate the number of occurrences of a given element. In our case, we assumed the data were distributed according to the Zipf law, which gives us directly the error distribution for a given element, if we know its rank. Thus, this allows performing an MLE on the rank of our elements, by returning $argmax_r P(\widehat{\eta}_x = G \mid x = x_r)$ using the following formula for each hash function,

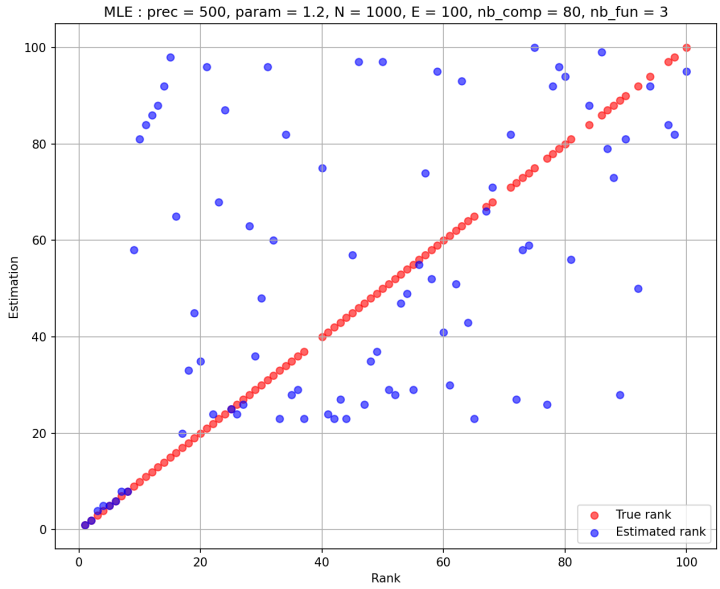

Figure 10: MLE on Count Min. "prec" is the number of iterations in the Monte Carlo, "nb_comp" is the number of counters per hash function and "nb_fun" is the number of hash functions. The holes in the red line correspond to ranks that were not in the stream, because their corresponding element has not been generated (see code for more detail).

approximating the situation by supposing each row of the sketch is independant:

Let E be the number of distinct elements from which we generate the stream (ie. we use Zipf(.,E)). Let $G \in \mathbb{N}$, $r \in [|1, E|]$, $t \in [|1, k|]$.

$$P(\widetilde{h_t}(x) = G \mid x = x_r) = \sum_{i=0}^{G} P(\eta_x = i \mid x = x_r) \cdot P(\widetilde{h_t}(x) = G \mid \eta_x = i, x = x_r) \tag{13}$$

$$= \sum_{i=0}^{G} \underbrace{\binom{N}{i} \left(\frac{\alpha}{r^z}\right)^i \left(1 - \frac{\alpha}{r^z}\right)^{N-i}}_{=:\beta_i \text{ (binomial)}} \cdot \underbrace{P(\widetilde{h_t}(x) = G \mid \eta_x = i, x = x_r)}_{=:\gamma_i \text{ (binomial on random subset)}} \tag{14}$$

$$= \sum_{i=0}^{G} \beta_i \cdot \left(\sum_{I \subset \mathcal{P}([|1,E|] \setminus \{r\})} P\left(I = \{x' \in \widetilde{S} : h_t(x) = h_t(x')\}\right) \cdot \right.$$
$$\left. \cdot P\left(|\{x' \in S : x' \in I\}| = G - i\right)\right) \tag{15}$$

$$= \sum_{i=0}^{G} \beta_i \cdot \left(\sum_{I \subset \mathcal{P}([|1,E|] \setminus \{r\})} \left(\frac{1}{m}\right)^{|I|} \left(1 - \frac{1}{m}\right)^{E-1-|I|} \cdot \right.$$
$$\left. \cdot \binom{N-i}{G-i} \left(\sum_{s \in I} \frac{\alpha}{s^z}\right)^{G-i} \left(1 - \sum_{s \in I} \frac{\alpha}{s^z}\right)^{N-G} \right) \tag{16}$$

The formula for β_i comes from the counting of the number of occurences of x_r (it is a binomial). The formula for γ_i comes from a conditioning on which elements have a collision with x_r, and then the counting of the number of occurences of those elements in S (binomial).

The difficulty is that γ_i is not computable in a reasonable time, as it contains $O(2^E)$ terms. To estimate this term, I used a Monte Carlo method.

But even with Monte Carlo, a lot of computing power was needed to obtain satisfactory results, so my algorithm yielded mixed results (see figure 10).

4.3 Hot & Cold Case Analysis

4.3.1 Hot & Cold Model

The idea of the Hot & Cold model came from the will to simplify the situation, in order to be able to analyse mathematically the situation and find computable algorithms.

In the Hot & Cold model, the input stream contains two types of elements: "hot" elements, which have a high frequency of occurrence, and "cold" elements, which have a lower frequency.

Let $\mathcal{H}$ be the set of hot elements, and $\mathcal{C}$ the set of cold elements. Let p_h be the probability that an element from the stream is hot, and p_c its probability of being cold ($p_h + p_c = 1$). We also fix $|\mathcal{H}|$ and $|\mathcal{C}|$, and we draw the elements of the stream one by one, first sampling from $\mathcal{H}$ and $\mathcal{C}$ according to a Bernoulli distribution with parameter p_h, and then uniformly selecting an element from the chosen set.

Thus, for a fixed element $x \in \mathcal{H}$, its probability of being the i^{th} element of S (with $i \in [|1, |S||]$) is $C_h := \frac{p_h}{|\mathcal{H}|}$ (we also define $C_c := \frac{p_c}{|\mathcal{C}|}$).

4.3.2 Multinoullis in a Sum Indexed by a Poisson Law

Property 1. *Let $r \in \mathbb{N}$, $N \sim \mathcal{P}(\lambda)$, $(X_n)_n$ i.i.d., with distribution $Cat(p_1, ..., p_r)$ ("Categorical distribution", also called Multinoulli. It is similar to Bernoulli, but with r possible outcomes instead of 2, with respective probabilities p_i), and taking values in $[|1, r|]$. For $j \in [|1, r|]$, let $S_j := \sum_{i=1}^{N} \mathbb{1}_{\{X_i = j\}}$.*
Then the $(S_j)_j$ are independent, of respective laws $\mathcal{P}(\lambda p_j)$.

4.4 Analysis in Our Setting

4.4.1 Principle

Let $N >> 1$. We consider a Count Sketch (CS) with n rows and m counters per row, and a stream S generated according to a hot & cold model with parameters $(\lambda_h, \lambda_c, p_h, p_c)$, of total length $|S|$ with $|S| \sim \mathcal{P}(N)$.

Let C_h (resp. C_c) be the probability of drawing a given hot (resp. cold) item when generating the stream. Denote $\mathcal{H}$ (resp. $\mathcal{C}$) the set of hot (resp. cold) items.

We have:

$$\begin{cases} C_h = \frac{p_h}{|\mathcal{H}|} = \frac{p_h}{\lambda_h m} \\ C_c = \frac{p_c}{|\mathcal{C}|} = \frac{p_c}{\lambda_c m} \end{cases}$$

The estimation of η_x on row i is given by:

$$\widehat{\eta}_{x,i} := s_i(x) \cdot \widetilde{h_i}(x) = \eta_x + \varepsilon_i(x)$$

We are interested in the distribution of the error term $\varepsilon_i(x)$. Without loss of generality, we assume $x \in \mathcal{H}$. Then:

$$\varepsilon_i(x) = \underbrace{\sum_{x' \in \mathcal{H} \setminus \{x\}} \underbrace{s_i(x)s_i(x') \cdot \mathbb{1}_{\{h_i(x)=h_i(x')\}} \cdot \eta_{h,x'}}_{:=X_{h,x'}}}_{:=\varepsilon_h} + \underbrace{\sum_{x' \in \mathcal{C}} \underbrace{s_i(x)s_i(x') \cdot \mathbb{1}_{\{h_i(x)=h_i(x')\}} \cdot \eta_{c,x'}}_{:=X_{c,x'}}}_{:=\varepsilon_c}$$

where:

$$\begin{cases} s_i(x)s_i(x') \sim \text{Rademacher} \\ \mathbb{1}_{\{h_i(x)=h_i(x')\}} \sim \text{Bernoulli}(1/m) \\ |S| \sim \mathcal{P}(N) \\ \eta_{h,x'} := \sum_{j=1}^{|S|} \mathbb{1}_{\{S[j]=x'\}} \sim \mathcal{P}(NC_h) \\ \eta_{c,x'} := \sum_{j=1}^{|S|} \mathbb{1}_{\{S[j]=x'\}} \sim \mathcal{P}(NC_c) \end{cases}$$

Moreover, from Property 1 (using "Poissonization"), the variables $\eta_{h,x'}$ and $\eta_{c,x'}$ are mutually independent.

Thus, using the Central Limit Theorem, we can approximate both ε_h and ε_c by Gaussian distributions.

4.5 Parameter Calculation

For this, using Poisson's moments:

$$\mathbb{E}(X_h, x') = 0 \tag{17}$$
$$\mathbb{V}(X_h, x') = \mathbb{E}((X_{h,x'})^2) \tag{18}$$
$$= \mathbb{E}(\mathbb{1}_{\{h_i(x)=h_i(x')\}})\mathbb{E}((\eta_h, x')^2) \tag{19}$$
$$= \frac{1}{m}(NC_h(NC_h + 1)) \tag{20}$$

We deduce:

$$\begin{cases} \mathbb{V}_h := \mathbb{V}(X_h, x') = \frac{1}{m}(NC_h(NC_h + 1)) \\ \mathbb{V}_c := \mathbb{V}(X_c, x') = \frac{1}{m}(NC_c(NC_c + 1)) \end{cases}$$

Thus, we have the approximations (still assuming $x \in \mathcal{H}$),

$$\begin{cases} \varepsilon_h \sim \mathcal{N}(0, \mathbb{V}_h(|\mathcal{H}| - 1)) \\ \varepsilon_c \sim \mathcal{N}(0, \mathbb{V}_c|\mathcal{C}|) \end{cases}$$

and finally,

$$\varepsilon_i(x) \sim \mathcal{N}(0, \underbrace{\mathbb{V}_h(|\mathcal{H}| - 1) + \mathbb{V}_c|\mathcal{C}|}_{=: \sigma_h^2}) \tag{21}$$

We also define $\sigma_c^2 := \mathbb{V}_h|\mathcal{H}| + \mathbb{V}_c(|\mathcal{C}| - 1)$.

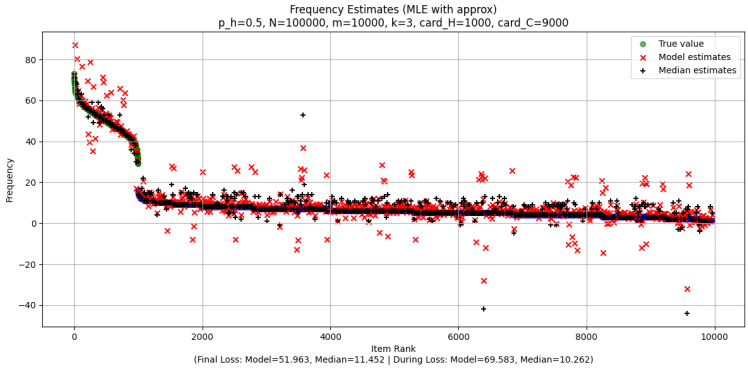

Figure 11: MLE with approx, $p_h = 0.5, N = 100000, m = 10000, k = 3, card_H = 1000, card_C = 9000$

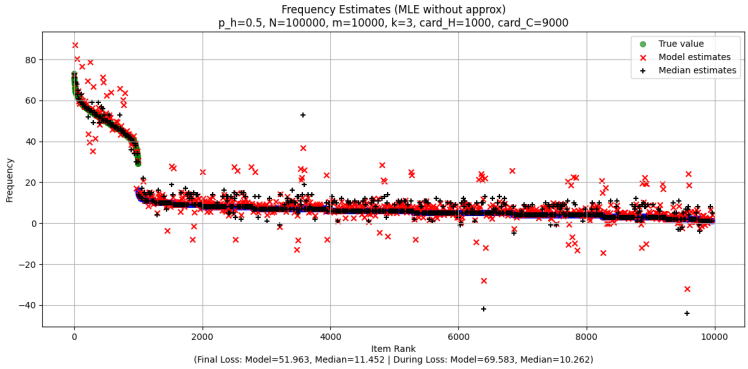

Figure 12: MLE without approx, $p_h = 0.5, N = 100000, m = 10000, k = 3, card_H = 1000, card_C = 9000$

4.6 Use of the Results

Now that the error distribution is known, we can perform MLE or MAP. We will consider we want to estimate the occurences η_x as the element arrive in the stream S. Ie. we will always want to estimate occurences of the form $\eta^j_{S[j]}$.

4.6.1 MLE on η_x

The MLE on η_x returns:

$$\widehat{\eta}_x := argmax_{\theta \in \mathbb{N}} \left[\sum_{i=1}^{n} log(f(\widehat{\eta}_{x,i} - \theta)) \right] \tag{22}$$

where f is the probability density function of $\varepsilon_1(x)$:

$$f : x \mapsto p_h \underbrace{\frac{1}{\sigma_h \sqrt{2\pi}} e^{\frac{-x^2}{2\sigma_h^2}}}_{=: f_h(x)} + p_c \underbrace{\frac{1}{\sigma_c \sqrt{2\pi}} e^{\frac{-x^2}{2\sigma_c^2}}}_{=: f_c(x)} \tag{23}$$

We note that

$$\begin{cases} f_h = f(\cdot | x \in \mathcal{H}) \\ f_c = f(\cdot | x \in \mathcal{C}) \end{cases}$$

If we make the approximation that $|\mathcal{H}| - 1 \approx |\mathcal{H}|$ and $|\mathcal{C}| - 1 \approx |\mathcal{C}|$, i.e. $\sigma_h \approx \sigma_c := \sigma$, then (since $p_h + p_c = 1$):

$$f : x \mapsto \frac{1}{\sigma_h \sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \tag{24}$$

Thus,

$$\widehat{\eta}_x = argmax_{\theta \in \mathbb{N}} \left[-\sum_{i=1}^{n} (\widehat{\eta}_{x,i} - \theta)^2 \right] \tag{25}$$

By setting the derivative to zero, we find

$$\widehat{\eta}_x = \frac{1}{n} \sum_{i=1}^{n} \widehat{\eta}_{x,i} \tag{26}$$

The MLE in this approximation therefore returns the mean of the values in the sketch!

To estimate the precision of my models against the precision of the median estimator, I used two "loss" functions:

loss_final:

$$\frac{\sum_{x \in S} (\eta_x - \widehat{\eta}_x)^2}{|S|} \tag{27}$$

and loss_during:

$$\frac{\sum_{j=|S|-9999}^{|S|} (\eta^j_{S[j]} - \widehat{\eta}^j_{S[j]})^2}{10000} \tag{28}$$

loss_final represent the use case where we read the entire stream and then try to estimate the occurences of all the elements we saw. I found it interesting, but in order to have good estimators for this, I should have replaced p_h by $\frac{|\mathcal{H}|}{|\mathcal{H}|+|\mathcal{C}|}$ and p_c by $\frac{|\mathcal{C}|}{|\mathcal{H}|+|\mathcal{C}|}$.

loss_during represent the real use case of the MLE and MAP estimators, when we want to estimate the occurences of the element as they come. I kept only the 10000 last elements, because it is the values which interest us and is faster to compute.

(see fig 11 and fig 12 for iterations of the final estimations of the algorithm, and the table in appendix for performance).

We can see in the figures that some estimations are negative. That is obviously impossible, so in a real use of those algorithms, one would try to keep the estimations positive or null.

4.6.2 MLE on {hot, cold}

The MLE on {hot, cold} returns:

$$\widehat{c}_x := argmax_{\theta \in \{hot,cold\}} \left[\sum_{i=1}^{n} log(g(\widehat{\eta}_{x,i}, \theta)) \right] \tag{29}$$

where

$$g : (y, z) \mapsto \begin{cases} \sum_{j=0}^{|S|} \mathbb{P}(\eta_x = j|hot) \cdot f_h(y - j) & \text{if } z = hot \\ \sum_{j=0}^{|S|} \mathbb{P}(\eta_x = j|cold) \cdot f_c(y - j) & \text{if } z = cold \end{cases} \tag{30}$$

To simplify calculations, one could consider that

$$\begin{cases} P(\eta_x = j|hot) = \mathbb{1}_{\{\ j=\mu_h\}} \\ P(\eta_x = j|cold) = \mathbb{1}_{\{\ j=\mu_c\}} \end{cases}$$

We then have

$$g : (y, z) \mapsto \begin{cases} f_h(y - \mu_h) & \text{if } z = hot \\ f_c(y - \mu_c) & \text{if } z = cold \end{cases} \tag{31}$$

and the MLE will amount to evaluating whether η_x is better estimated by μ_c or μ_h, which seems natural.

The loss functions here become:

loss_final:

$$\frac{\sum_{x \in S} \mathbb{1}_{\{class\_x = estimated\_class\_x\}}}{|S|} \tag{32}$$

and loss_during:

$$\frac{\sum_{x \in S[-1000:]} \mathbb{1}_{\{class\_x = estimated\_class\_x\}}}{1000} \tag{33}$$

where $S[-1000:]$ is the set of the 1000 last elements of S.

(see table in appendix for performance).

4.6.3 MAP on η_x

To get the MAP, we multiply by the prior π, which can be approximated by a mixture of two Gaussians. The final density will be the product of the mixture of two Gaussians density by a Gaussian density.

Indeed, the MAP on η_x returns:

$$\widehat{\eta}_x := argmax_{\theta \in \mathbb{N}} \left[\log(\pi(\theta)) + \sum_{i=1}^{n} \log(f(\widehat{\eta}_{x,i} - \theta)) \right] \tag{34}$$

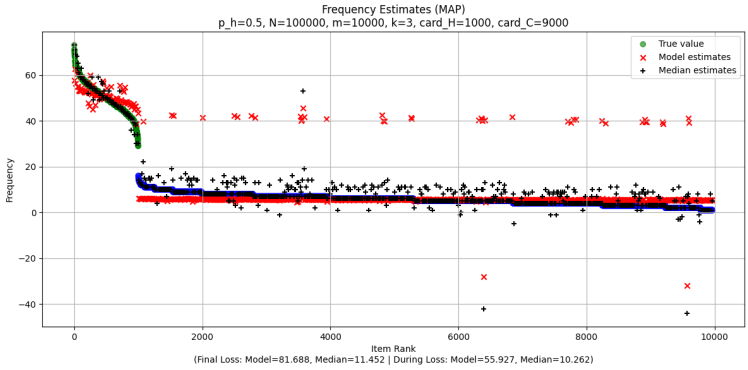

Figure 13: MAP, $p_h = 0.5, N = 100000, m = 10000, k = 3, card_H = 1000, card_C = 9000$

where

$$\pi : y \mapsto p_h \frac{1}{\sigma'_h \sqrt{2\pi}} e^{\frac{-(y-\mu_h)^2}{2\sigma'^2_h}} + p_c \frac{1}{\sigma'_c \sqrt{2\pi}} e^{\frac{-(y-\mu_c)^2}{2\sigma'^2_c}} \tag{35}$$

with

$$\begin{cases} \mu_h = C_h N \\ \sigma'^2_h = C_h N \\ \mu_c = C_c N \\ \sigma'^2_c = C_c N \end{cases}$$

according to the independence given by property 1. We approximate here the poisson distribution by a normal distribution, to simplify the optimization and avoid overflow errors. Optimization methods such as Newton's method, Nelder-Mead, or BFGS can be used to maximize this function.

(see fig 13 for an iteration of the final estimations of the algorithm, and the table in appendix for performance).

4.6.4 MAP on $\{hot, cold\}$

The MAP on $\{hot, cold\}$ returns:

$$\widehat{c}_x := argmax_{\theta \in \{hot, cold\}} \left[\log(\pi(\theta)) + \sum_{i=1}^{n} \log(g(\widehat{\eta}_{x,i}, \theta)) \right] \tag{36}$$

where

$$\pi : y \mapsto \begin{cases} p_h & \text{if } \theta = \text{hot} \\ p_c & \text{if } \theta = \text{cold} \end{cases} \tag{37}$$

(see table in appendix for performance).

As we can see in the appendix, our estimators work better than the median estimator on some type of streams, especially when we do not have much information in the sketch. I then tried to extend the results of this analysis to the original problem, and the case where the stream is generated with a Zipf distribution (see appendix).

5 Conclusion

This internship focused on investigating whether the performance of sketching algorithms for frequency estimation could be improved by leveraging prior knowledge of the stream's underlying data distribution. Our work shows that this is indeed a promising direction.

5.1 Summary of Contributions

The central achievement of this work was the detailed theoretical analysis of the Count-Sketch algorithm in a "Hot & Cold" generative model. By modeling the error term as a sum of independent random variables, we approximated its distribution with a Gaussian mixture. We successfully derived and implemented Maximum Likelihood (MLE) and Maximum A Posteriori (MAP) estimators that incorporate the known parameters of the Hot & Cold model, achieving better "on the fly" estimations than with the median estimator on some type of streams, here when we do not have much information about the stream in the sketch. The analysis above makes us believe our algorithm would be even better on bigger streams. Furthermore, we conducted a preliminary theoretical investigation into extending this analysis to streams governed by a Zipf distribution. This analysis, based on Lyapunov's condition for the Central Limit Theorem, revealed that a similar Gaussian error approximation is possible, but only under restrictive conditions on the distribution's parameters and the sketch's size.

5.2 Limitations and Comparison with Objectives

The primary objective of developing a more precise estimator for a known distribution was met in some cases for the Hot & Cold model. However, our work has several limitations. The analysis relies on a series of approximations, most notably the Gaussian nature of the error, which holds asymptotically. We did not investigate in details where our algorithm was better than the median or not. The theoretical extension to the Zipf model was not experimentally validated due to time constraints and the complexity of the conditions required. This analysis suggests that the approach is most effective for Zipf distributions that are relatively close to uniform ($z < 1/2$) and in a supercritical regime where the sketch has few counters, a scenario that may limit practical applicability.

5.3 Perspectives for Future Work

This research opens several avenues for future exploration. The most direct next steps would be to experimentally implement and validate the theoretical analysis for the Zipf case to quantify the practical gains under the identified constraints, as well as analyze more precisely the performance of the MLE and MAP estimators presented here, compared with the performance of the median estimator. A more ambitious direction would be to develop adaptive algorithms that first estimate the stream's distributional parameters (e.g., the Zipf exponent or the Hot & Cold parameters) and then dynamically apply the corresponding specialized estimator.

5.4 Overall experience

My internship at the LIGM was an enriching experience, combining technical learning with theoretical and experimental testing. I now have a better understanding of data sketching algorithms and research in general.

I am grateful to the researchers at the LIGM, especially my tutor, M. Kucherov, and M. Fuzy, for their warm welcome and guidance throughout my internship. I would also like to thank

Mme Palescandolo for her explanations, and M. Exibard for his kindness and hospitality. This experience helped shape my professional orientation and allowed me to discover new areas of research.

References

Lada A Adamic. Zipf, power-laws, and pareto-a ranking tutorial. *Xerox Palo Alto Research Center, Palo Alto, CA, http://ginger. hpl. hp. com/shl/papers/ranking/ranking. html*, 2000.

Patrick Billingsley. *Probability and Measure*. Wiley-Interscience, New York, 3rd edition, 1995.

Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

Peiqing Chen, Yuhan Wu, Tong Yang, Junchen Jiang, and Zaoxing Liu. Precise error estimation for sketch-based flow measurement. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 113–121, 2021.

Saar Cohen and Yossi Matias. Spectral bloom filters. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 241–252, 2003.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005a.

Graham Cormode and Shan Muthukrishnan. Summarizing and mining skewed data streams. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 44–55. SIAM, 2005b.

Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM transactions on networking*, 8(3):281–293, 2000.

Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science*, (Proceedings), 2007.

Éric Fusy and Gregory Kucherov. Analysis of count-min sketch under conservative update. *CoRR*, 2022.

Younes Ben Mazziane, Sara Alouf, and Giovanni Neglia. Analyzing count min sketch with conservative updates. *Computer Networks*, 217:109315, 2022.

Gregory T Minton and Eric Price. Improved concentration bounds for count-sketch. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 669–686. SIAM, 2014.

Moses Charikar Scribe and Carson Kent. Lecture 8: Cr-precis and count sketch.

Daniel Ting. Count-min: Optimal estimation and tight error bounds using empirical error distributions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2319–2328, 2018.

6 Appendix

6.1 Experiments on the Hot & Cold Model - performances

Here are the performance we found for the Hot & Cold analysis by taking the mean of 3 iterations of the different algorithms with the same parameters $p_h = 0.5, N = 100000, m = 10000, k = 3, |\mathcal{H}| = 1000, |\mathcal{C}| = 9000$:

	loss_final	**loss_during**
Median on η_x	11.452	10.262
MLE on η_x with approx	51.963	69.583
MLE on η_x without approx	51.963	69.583
MAP on η_x	81.688	55.927
MLE on $\{hot, cold\}$ with approx	1.0%	2.1%
MLE on $\{hot, cold\}$ without approx	1.3%	2.1%
MAP on $\{hot, cold\}$ with approx	1.0%	2.1%
MAP on $\{hot, cold\}$ without approx	1.3%	2.1%

And the mean on 3 iterations using the parameters $p_h = 0.5, N = 100000, m = 10000, k = 3, |\mathcal{H}| = 10000, |\mathcal{C}| = 90000$:

	loss_final	**loss_during**
Median on η_x	13.924	10.537
MLE on η_x with approx	10.511	8.216
MLE on η_x without approx	10.511	8.216
MAP on η_x	5.798	4.957
MLE on $\{hot, cold\}$ with approx	32.3%	28.3%
MLE on $\{hot, cold\}$ without approx	32.2%	29.0%
MAP on $\{hot, cold\}$ with approx	32.3%	28.3%
MAP on $\{hot, cold\}$ without approx	32.2%	29.0%

Even though this is not a complete analysis of performance, we can see here that the MLE, and especially the MAP estimator seem better than the Median estimator on η_x in the second case. When analysing the situation, it is a case where the elements do not have many occurences, and where the sketch contains few informations about the elements considered (see fig 14 and fig 15).

In the first case, the median estimator performs better (it corresponds to figs 11, 12 and 13).

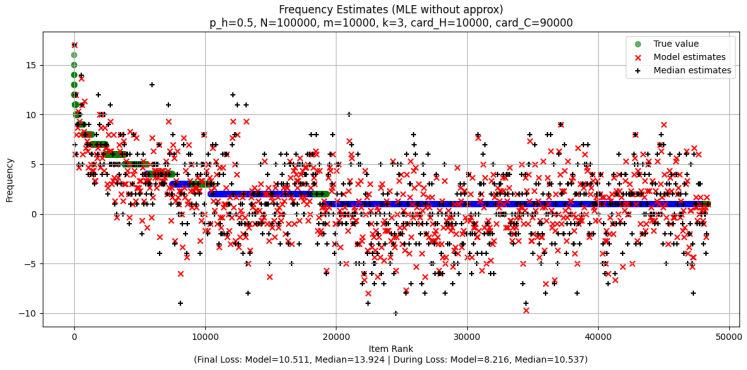

Figure 14: MLE without approx, $p_h = 0.5, N = 100000, m = 10000, k = 3, card_H = 10000, card_C = 90000$

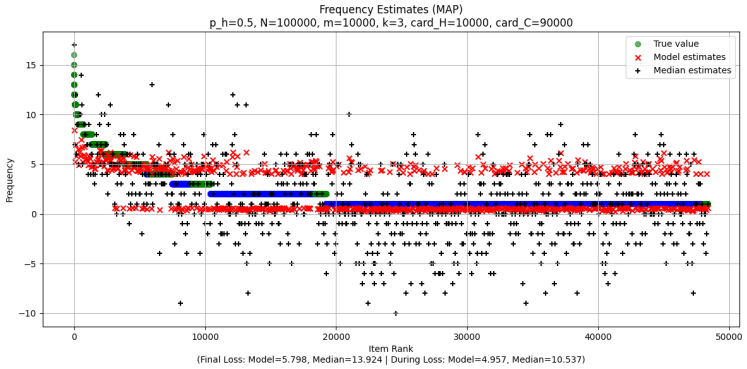

Figure 15: MAP, $p_h = 0.5, N = 100000, m = 10000, k = 3, card_H = 10000, card_C = 90000$

A strong hypothesis for the difference in performance in both cases is that when the signal-to-noise ratio is low (many distinct elements, few occurrences per element, as in the second case), the prior knowledge encoded in the MAP and MLE estimator provides a significant advantage over the median, which is more susceptible to noise. Conversely, when the signal is strong (fewer elements, higher frequencies), the median estimator is robust and sufficient, and the model assumptions or approximations in the new estimators might introduce their own errors.

We also see that the approximation on the MLE is reasonable in both cases.

The sometimes better performance of MLE and MAP with loss_during is no surprise, as their formulas are made to estimate the occurences "on the fly", which corresponds to the formula of loss_during. The sometimes better performance of all the estimators with loss_during can be explained by the repetition of some "easy to estimate" elements.

For the analysis on $\{hot, cold\}$, we can see the estimators do a "pretty good job" at distinguishing the two classes. Also, the MAP estimator gives the same results as the MLE because $p_h = \frac{1}{2}$.

We can note that the loss is higher when the occurences of our different elements in S are higher, because the differences between the estimations and the true values grows.

6.2 Proofs

Proof of property 1.

Lemma 1. *With the notations of property 1:* $\forall\, j \in \mathbb{N}^*, S_j \sim \mathcal{P}(\lambda\ p_j)$.

Proof of lemma 1. Let $j \in \mathbb{N}^*, k \in \mathbb{N}$.

$$P(S_j = k) = P(\sum_{i=1}^{N} \mathbb{1}_{\{X_i=j\}} = k) \tag{38}$$

$$= \sum_{s=k}^{+\infty} P(N = s) \cdot P(\sum_{i=1}^{s} \mathbb{1}_{\{X_i=j\}} = k) \tag{39}$$

$$= \sum_{s=k}^{+\infty} \frac{e^{-\lambda}\lambda^s}{s!} \cdot \binom{s}{k} p_j^k (1-p_j)^{s-k} \tag{40}$$

$$= e^{-\lambda} p_j^k \frac{\lambda^k}{k!} \sum_{s=k}^{+\infty} \frac{(\lambda(1-p_j))^{s-k}}{(s-k)!} \tag{41}$$

$$= e^{-\lambda p_j} \frac{(\lambda p_j)^k}{k!} \tag{42}$$

□

Let $n \in \mathbb{N}$, $i_1 < i_2 < ... < i_n \in \mathbb{N}^*$, and $\forall\, j \leq n,\ k_j \in \mathbb{N}$. We have:

$$\begin{aligned} P\left(\bigcap_{j=1}^{n}(S_{i_j} = k_j)\right) &= \sum_{s=\sum_{l=1}^{n} k_l}^{+\infty} P(N = s) \cdot P(S_{i_1} = k_1 \mid N = s) \cdot P(S_{i_2} = k_2 \mid S_{i_1} = k_1, N = s) \cdot \\ &\quad \cdots \cdot P(S_{i_n} = k_n \mid S_{i_1} = k_1, \ldots, S_{i_{n-1}} = k_{n-1}, N = s) \\ &= \sum_{s=\sum_{l=1}^{n} k_l}^{+\infty} \frac{e^{-\lambda}\lambda^s}{s!} \cdot \binom{s}{k_1} p_1^{k_1} (1-p_1)^{s-k_1} \cdot \\ &\quad \cdot \binom{s-k_1}{k_2} \left(\frac{p_2}{1-p_1}\right)^{k_2} \left(1 - \frac{p_2}{1-p_1}\right)^{s-k_1-k_2} \cdot \\ &\quad \cdot \cdots \cdot \binom{s - \sum_{l=1}^{n-1} k_l}{k_n} \left(\frac{p_n}{1 - \sum_{l=1}^{n-1} p_l}\right)^{k_n} \cdot \\ &\quad \cdot \left(1 - \frac{p_n}{1 - \sum_{l=1}^{n-1} p_l}\right)^{s - \sum_{l=1}^{n} k_l} \end{aligned} \tag{43}$$

We have two telescoping products, one on the probabilities and one on the factorial terms in the binomial coefficients. Let us simplify them.

Let $K := \sum_{l=1}^{n} k_l$. Then:

$$P\left(\bigcap_{j=1}^{n}(S_{i_j}=k_j)\right) = \sum_{s=K}^{+\infty}\frac{e^{-\lambda}\lambda^s}{s!}\cdot\frac{s!}{k_1!\cdot k_2!\cdots k_n!\cdot(s-K)!}\cdot p_1^{k_1}\cdots p_n^{k_n}\cdot(1-\sum_{j=1}^{n}p_j)^{s-K} \quad (44)$$

$$= \frac{p_1^{k_1}\cdots p_n^{k_n}}{k_1!\cdots k_n!}\cdot\sum_{s=K}^{+\infty}\frac{e^{-\lambda}\lambda^s}{(s-K)!}\cdot(1-\sum_{j=1}^{n}p_j)^{s-K} \quad (45)$$

$$= \frac{p_1^{k_1}\cdots p_n^{k_n}}{k_1!\cdots k_n!}\cdot\lambda^K e^{-\lambda}\cdot\sum_{u=0}^{+\infty}\frac{(\lambda(1-\sum_{j=1}^{n}p_j))^u}{u!} \quad (46)$$

$$= \frac{p_1^{k_1}\cdots p_n^{k_n}}{k_1!\cdots k_n!}\cdot\lambda^K e^{-\lambda\sum_{j=1}^{n}p_j} \quad (47)$$

$$= \prod_{j=1}^{n}\left(\frac{(\lambda p_j)^{k_j}e^{-\lambda p_j}}{k_j!}\right) \quad (48)$$

Thus, the random variables $(S_j)_{j\in\mathbb{N}}$ are independent and each follows a Poisson distribution with parameter λp_j.

□

6.3 Generalization and extension of the Hot & Cold analysis to the Zipf case

The main difference between the Hot & Cold problem and the original problem is that there are many elements (on the order of E) per class in the Hot & Cold problem, which allows the use of the Central Limit Theorem (CLT) to estimate the error with a Gaussian.

In the original problem, our stream is drawn according to the Zipf distribution, and thus if we write $|S| \sim \mathcal{P}(\mathrm{N})$, we will have independent variables (property 1), but not identically distributed. We loosely conflate an element and its rank in the sequence (ie. $k := x_k$, $\eta_k := \eta_{x_k}$).

However, there exists an extension of the CLT in the case of independent but non-identically distributed variables:

Theorem 3. *Lindeberg's Condition - cf. Wikipedia*

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and $X_k : \Omega \to \mathbb{R}$, $k \in \mathbb{N}$ be independent random variables on this space. Suppose the expectations $\mathbb{E}[X_k] = \mu_k$ and variances $\mathrm{Var}[X_k] = \sigma_k^2$ exist and are finite. Also, let $s_n^2 := \sum_{k=1}^{n}\sigma_k^2$.

If the sequence (X_k) satisfies the "Lindeberg condition":

$$\lim_{n\to\infty}\frac{1}{s_n^2}\sum_{k=1}^{n}\mathbb{E}\left[(X_k-\mu_k)^2\cdot\mathbb{1}_{\{|X_k-\mu_k|>\varepsilon s_n\}}\right] = 0 \quad (49)$$

for all $\varepsilon > 0$, then the CLT holds, i.e., the random variables

$$Z_n := \frac{\sum_{k=1}^{n}(X_k-\mu_k)}{s_n} \quad (50)$$

converge in distribution, as $n \to \infty$, to a standard normal distribution.

Remark 1. *Lindeberg's condition is sufficient, but generally not necessary. However, if the sequence of independent random variables satisfies*

$$\max_{k=1,\ldots,n} \frac{\sigma_k^2}{s_n^2} \to 0, \quad \text{as } n \to \infty, \tag{51}$$

then Lindeberg's condition is both necessary and sufficient.

There exists a weaker, yet more manageable, condition:

Theorem 4. *Lyapunov's Condition - cf. Wikipedia*

Let $(X_n)_{n\geq 1}$ be a sequence of independent random variables defined on the same probability space. Suppose that for $n \geq 1$, X_n has finite expectation μ_n and finite standard deviation σ_n, and define:

$$s_n^2 = \sum_{i=1}^{n} \sigma_i^2$$

and

$$Z_n = \frac{1}{s_n} \sum_{i=1}^{n} (X_i - \mu_i).$$

Suppose that for some $\delta > 0$, the Lyapunov condition is satisfied:

$$\lim_{n\to+\infty} \frac{1}{s_n^{2+\delta}} \sum_{i=1}^{n} \mathbb{E}\left[|X_i - \mu_i|^{2+\delta}\right] = 0,$$

then Lindeberg's condition is fulfilled.

Thus, by the previous theorem, the normalized sum of the X_i converges to a standard normal distribution:

$$Z_n \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

In fact, we will even need X_i to depend on n, that is, to deal with a triangular array. There exists an extension of Lyapunov's condition for triangular arrays:

Theorem 5. *Lyapunov for triangular arrays - cf. Billingsley (1995)*

Let, for each integer $n \geq 1$, a family of independent random variables

$$X_{n1}, X_{n2}, \ldots, X_{nr_n}$$

be defined on a probability space (possibly depending on n). Assume that:

- *each X_{nk} has zero mean and finite variance:*

$$\mathbb{E}[X_{nk}] = 0, \quad \mathrm{Var}(X_{nk}) = \sigma_{nk}^2;$$

- *the total variance of the sum*

$$S_n := \sum_{k=1}^{r_n} X_{nk}$$

is denoted

$$s_n^2 := \sum_{k=1}^{r_n} \sigma_{nk}^2 > 0;$$

- *there exists a real number $\delta > 0$ such that the $(2+\delta)$-th moments are finite and Lyapunov's condition is verified:*

$$\lim_{n\to\infty} \frac{1}{s_n^{2+\delta}} \sum_{k=1}^{r_n} \mathbb{E}\left[|X_{nk}|^{2+\delta}\right] = 0.$$

Then, the normalized sum S_n/s_n converges in distribution to a standard normal distribution:

$$\frac{S_n}{s_n} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1).$$

6.3.1 Analysis, application of the Lyapunov condition to our case

I use the Lyapunov condition for triangular arrays, since I will make $N := |S|$ and m depend on E.

We take $z < \frac{1}{3}$ and N such that $NE^{-1} \xrightarrow[E\to+\infty]{} +\infty$. We adapt the notations from the Hot & Cold case. We fix an element k_0. We set

$$\begin{cases} X_k := s_i(k_0)s_i(k) \cdot \mathbb{1}_{\{h_i(k_0)=h_i(k)\}} \cdot \eta_k \\ \alpha := \frac{1}{\sum_{k=1}^{E} \frac{1}{k^z}} \end{cases}$$

And the error due to an element k when studying k_0 is:

$$\varepsilon_{i,k_0} = \sum_{k\neq k_0} X_k$$

Later on, I could approximate ε_{i,k_0} by $\sum_k X_k$. I did not do so, but removing the element of rank k_0 only adds negligible terms compared to the others, as can be seen in lines (58) and (63).

We have

$$\sum_{k=1}^{n} \frac{1}{k^p} \sim \begin{cases} \frac{n^{1-p}}{1-p} & \text{if } 0 < p < 1, \\ \ln n & \text{if } p = 1, \\ \zeta(p) & \text{if } p > 1. \end{cases}$$

therefore, since $z < \frac{1}{3}$:

$$\alpha^2 \sum_{k=1}^{E} \frac{1}{k^{2z}} \sim \frac{(1-z)^2}{1-2z} \cdot \frac{E^{1-2z}}{E^{2-2z}} =: C_2 E^{-1} \tag{52}$$

$$\alpha^3 \sum_{k=1}^{E} \frac{1}{k^{3z}} \sim \frac{(1-z)^3}{1-3z} \cdot \frac{E^{1-3z}}{E^{3-3z}} =: C_3 E^{-2} \tag{53}$$

$\eta_k \sim \mathcal{P}(N\frac{\alpha}{k^z})$, hence:

$$s_n^2 = \sum_{\substack{k=1 \\ k\neq k_0}}^{E} \mathbb{V}(X_k) \tag{54}$$

$$= \left(\sum_{k=1}^{E} \frac{1}{m}\mathbb{E}(\eta_k^2)\right) - \frac{1}{m}\mathbb{E}(\eta_{k_0}^2) \tag{55}$$

$$= \left(\sum_{k=1}^{E} \frac{1}{m}(N\frac{\alpha}{k^z}(1+N\frac{\alpha}{k^z}))\right) - \frac{1}{m}(N\frac{\alpha}{k_0^z}(1+N\frac{\alpha}{k_0^z})) \tag{56}$$

$$= \frac{1}{m}\left(N + \alpha^2N^2\sum_{k=1}^{E}\frac{1}{k^{2z}} - (N\frac{\alpha}{k_0^z}(1+N\frac{\alpha}{k_0^z}))\right) \tag{57}$$

$$= \frac{1}{m}\left(N + N^2C_2E^{-1} + o(N^2E^{-1})\right) \tag{58}$$

thus

$$s_n^3 = \frac{1}{m^{\frac{3}{2}}}\left(N + N^2C_2E^{-1} + o(N^2E^{-1})\right)^{\frac{3}{2}} \tag{59}$$

Then, taking $\delta = 1$,

$$\sum_{\substack{k=1 \\ k\neq k_0}}^{E} \mathbb{E}\left[|X_k|^3\right] = \frac{1}{m}\left(\sum_{k=1}^{E}\mathbb{E}\left[\eta_k^3\right]\right) - \mathbb{E}\left[\eta_k^3\right] \tag{60}$$

$$= \frac{1}{m}\sum_{k=1}^{E}\left(N\frac{\alpha}{k^z} + 3\left(N\frac{\alpha}{k^z}\right)^2 + \left(N\frac{\alpha}{k^z}\right)^3\right) - \left(N\frac{\alpha}{k_0^z} + 3\left(N\frac{\alpha}{k_0^z}\right)^2 + \left(N\frac{\alpha}{k_0^z}\right)^3\right) \tag{61}$$

$$= \frac{1}{m}\left(N + 3N^2\alpha^2\sum_{k=1}^{E}\frac{1}{k^{2z}} + N^3\alpha^3\sum_{k=1}^{E}\frac{1}{k^{3z}} + o(N^3E^{-2})\right) \tag{62}$$

$$= \frac{1}{m}\left(N + 3C_2N^2E^{-1} + o(N^2E^{-1}) + C_3N^3E^{-2} + o(N^3E^{-2})\right) \tag{63}$$

Hence,

$$\frac{1}{s_n^3}\sum_{k=1}^{E}\mathbb{E}\left[|X_k|^3\right] = \frac{\frac{1}{m}\left(N + 3C_2N^2E^{-1} + o(N^2E^{-1}) + C_3N^3E^{-2} + o(N^3E^{-2})\right)}{\frac{1}{m^{\frac{3}{2}}}\left(N + N^2C_2E^{-1} + o(N^2E^{-1})\right)^{\frac{3}{2}}} \tag{64}$$

$$= \frac{N + 3C_2N^2E^{-1} + o(N^2E^{-1}) + C_3N^3E^{-2} + o(N^3E^{-2})}{m^{-\frac{1}{2}}\left(N + N^2C_2E^{-1} + o(N^2E^{-1})\right)^{\frac{3}{2}}} \tag{65}$$

$$\sim \frac{C_3N^3E^{-2}}{C_2^{\frac{3}{2}}m^{-\frac{1}{2}}N^3E^{-\frac{3}{2}}} \quad \text{since } NE^{-1} \xrightarrow[E\to+\infty]{} +\infty \tag{66}$$

$$= \frac{C_3}{C_2^{\frac{3}{2}}} \cdot m^{\frac{1}{2}}E^{-\frac{1}{2}} \tag{67}$$

To satisfy the Lyapunov condition, it is therefore sufficient to have:

$$\begin{cases} NE^{-1} \xrightarrow[E\to+\infty]{} +\infty \\ mE^{-1} = o(1) \\ z < \frac{1}{3} \end{cases}$$

For example, we can take:

$$\begin{cases} N = E^2 \\ m = \sqrt{E} \\ z < \frac{1}{3} \end{cases}$$

The first condition seems reasonable, and tells us that we need a stream that grows faster than the number of distinct elements. The second condition is constraining, as it forces us to work in a supercritical regime, where the number of counters is very small compared to the number of distinct elements in the stream (the Count Sketch is likely to contain a lot of noise due to many collisions, and may be of little use). The third condition tells us that the Zipf distribution must be close to a uniform distribution — since $Zipf(0, N) = \mathbb{U}([|1, N|])$.

□

I chose to consider $z < \frac{1}{3}$ for convenience (to have only one case concerning the zeta equivalent), but similar results hold for $z \leq \frac{1}{2}$. However, if we consider $z > \frac{1}{2}$, the line (67) *will contain $= \frac{C_3}{C_2^{\frac{3}{2}}} \cdot m^{\frac{1}{2}}$, which cannot go to* 0.

Thus, given $NE^{-1} \xrightarrow[E\to+\infty]{} +\infty$, the Lyapunov condition can only be verified if $z \leq \frac{1}{2}$, and some other condition on m dependant of z.

If we remove the hypothesis $NE^{-1} \xrightarrow[E\to+\infty]{} +\infty$, the analysis becomes very different, and it does not seem easy to attain the Lyapunov condition.

6.3.2 Conclusion concerning the generalization

If we are in a case that satisfies the Lyapunov condition, and consequently the Lindeberg condition, we can then reason similarly to what was done in the Hot & Cold case to estimate the occurrences of the elements of the stream from the Count Sketch. This analysis shows us that this works if our Zipf distribution resembles a uniform distribution, which was expected. But also if we have a very large stream and relatively few counters compared to the number of distinct elements, which is more surprising.

I did not implement this during my internship due to time constraints.

6.4 Difficulties concerning the code

I did not elaborate on the subtleties in my code in this rapport, but I needed to find ways to have pairwise independant hash functions, to generate data following the Zipf law, to present my data in a readable way, find a way to numerically minimize a function (especially when it can take very small values), etc. This was not especially hard nor very technical, but it took time.

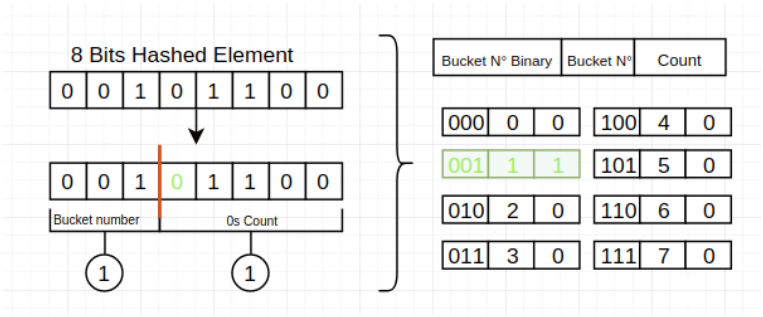

Figure 16: Hyperloglog. Source: Medium

6.5 Related subjects I encountered during my internship

Firstly, two data sketching algorithms very different from the ones I used during my internship:

6.5.1 Flajolet Martin Algorithm

The Flajolet-Martin (FM, Flajolet and Martin (1985)) algorithm is one of the earliest probabilistic methods for estimating the number of distinct elements (the *cardinality*) in a data stream, using very little memory (it is part of the Hyperloglog algorithm, shown in figure 16).

The key idea is that if we hash each element of the stream uniformly to binary strings, the probability that a hash ends with 2^r (1 followed by r zeros) is $1/2^{(r+1)}$. Therefore, if we observe a hash ending in a large number of zeros, this is evidence of many distinct elements having been seen.

Formally, let h be a hash function mapping elements of the stream to b-bit integers uniformly. For each incoming element x, we compute $h(x)$ and determine the number of trailing zeros in its binary representation, denoted $\rho(h(x))$. We then follow this algorithm:

Algorithm 3: Flajolet–Martin Algorithm

1: Initialize a bit-vector BITMAP to be of length L and contain all 0s.
2: **for** each element x in M **do**
3: Calculate the index $i = \rho(\text{hash}(x))$.
4: Set BITMAP$[i] = 1$.
5: **end for**
6: Let R denote the smallest index i such that BITMAP$[i] = 0$.
7: Estimate the cardinality of M as $2^R/\phi$, where $\phi \approx 0.77351$.

The calculations leading to ϕ can be found in the original article (Flajolet and Martin, 1985).

6.5.2 Hyperloglog

HyperLogLog (HLL, Flajolet et al. (2007)) is a highly optimized variant of the Flajolet-Martin algorithm that improves estimation accuracy while maintaining a low memory footprint. It is widely used in practice, notably in distributed systems and database engines, for cardinality estimation.

The core idea is to use a single hash function h mapping stream elements to long bit strings, and split each hash into two parts:

- The first p bits of $h(x)$ determine which of $m = 2^p$ registers will be updated.

- The remaining bits are used to compute the number of leading zeros, $\rho(w)$.

(see figure 16).

Each register $M[i]$ keeps track of the maximum number of leading zeros observed among all elements mapped to it:

$$M[i] := \max(M[i], \rho(w))$$

At the end of the stream, the estimate is computed using the harmonic mean of the $2^{-M[i]}$ values:

$$Z = \left(\sum_{i=1}^{m} 2^{-M[i]} \right)^{-1}, \quad \text{and} \quad \widetilde{n} = \alpha_m \cdot m^2 \cdot Z$$

Here, α_m is a correction constant depending on m. Additional corrections are applied for small and very large cardinalities to improve the estimator's robustness.

Compared to FM, HyperLogLog has much lower variance and can estimate large cardinalities (millions or more) using just a few kilobytes of memory. This makes it a practical and powerful tool in large-scale data processing pipelines.

Then, some things I found interesting:

6.5.3 Introduction to hypergraphs

Sketching algorithm can often be thought of as constructing specific types of hypergraphs.

A hypergraph H is a pair (V, E) where

$$V = \{v_1, v_2, \ldots, v_n\}$$

is a non-empty set (generally finite), and

$$E = \{E_1, E_2, \ldots, E_m\}$$

is a family of non-empty subsets of V.

Like graphs, we say that:

- The elements of V are the vertices of H.
- The number of vertices n is the order of the hypergraph.
- The elements of E are the edges of H.

Hypergraphs correspond precisely to matrices with coefficients 0 or 1 (where each column contains at least one 1).

Indeed, every hypergraph H corresponds uniquely to a matrix $\mathcal{A}_{n,m}$ such that:

$$\forall a_{i,j} \in \mathcal{A}, \quad a_{i,j} = \begin{cases} 1 & \text{if } v_i \in E_j \\ 0 & \text{otherwise} \end{cases}$$

In the sketches presented in this report, other than Flajolet-Martin and Hyperloglog, setting $\mathcal{E}$ as the set of stream elements and $h_1, ..., h_k$ the considered hash functions, one can construct the hypergraph associated to a sketch by setting:

- V is the set of cells of the sketch (the table in the case of the Bloom Filter, the matrix in Count Sketch or Count Min)

- n is the total number of counters/booleans
- $E := \{\{h_i(e)\}_{i=1}^k : e \in \mathcal{E}\}$

This brings interesting properties. For example, for CM, if there exists a vertex of degree 1 in its associated hypergraph, then a unique element hashes on this vertex and thus the estimated occurrence count for this vertex will be the correct occurrence count. There is the notion of a "peelable" graph, which is the class of graphs that can be "peeled", by iteratively removing a multi-edge connected to a vertex of degree 1 until the graph contains no edges. In such a graph, the peeling procedure would in theory allow recovering the true occurrences of all elements hashed in the data structure. But one needs to detect vertices of degree 1. I did not study these questions during my internship.

6.5.4 Introduction to the Dirichlet process

The Dirichlet Process is a data generation process widely used in Bayesian probability due to its good properties. It models data with the "rich-get-richer" property.

There are several ways to view it, but the simplest version (not necessarily the most natural) is the following:

First, fix a distribution H called the "base distribution", and a "concentration parameter" $\alpha \in \mathbb{R}_+^*$.

Then generate the stream $S := (X_n)_n$ using the following procedure:

For $n \geq 1$:

$$\begin{cases} \text{with probability } \frac{\alpha}{\alpha+(n-1)}, \text{ draw } X_n \text{ from } H. \\ \text{with probability } \frac{\eta_x}{\alpha+(n-1)}, \text{ set } X_n = x, \text{ where } \eta_x \text{ is the number of occurrences of } x \text{ in } (X_i)_{i<n}. \end{cases}$$

Here, the X_n are not independent, but they are exchangeable, meaning that for any permutation σ of $\mathbb{N}$, $(X_n)_n \sim (X_{\sigma(n)})_n$.

In this framework, we can apply De Finetti's representation theorem, which shows that there exists a distribution P such that the X_n are iid with distribution P. This distribution P is itself a random variable. Its law is called the "Dirichlet process" (DP). Thus, we obtain an equivalent of the above process as follows:

1. draw a distribution P according to $DP(H, \alpha)$
2. draw the X_n independently from P

This generative view is directly related to two other famous analogies for the Dirichlet Process: the Chinese Restaurant Process and the Stick-Breaking Construction.

The Chinese Restaurant Process (CRP) The "rich-get-richer" sampling scheme is a direct formulation of the Chinese Restaurant Process (CRP). In this metaphor, the data points $(X_n)_n$ are customers arriving at a restaurant with a potentially infinite number of tables.

- Each distinct value observed so far corresponds to an occupied table.
- When customer n arrives, they join an existing table x with a probability proportional to how many customers are already there (η_x). This corresponds to setting $X_n = x$.

- Alternatively, the customer sits at a new, unoccupied table with a probability proportional to the concentration parameter α. This corresponds to drawing a new value for X_n from the base distribution H.

The CRP describes the distribution over the *partitions* of the data points, while the Dirichlet Process describes the distribution over the (random) probability measure P from which the data is drawn.

The Stick-Breaking Construction The Stick-Breaking Construction provides a direct, constructive method for generating the random distribution $P \sim \mathrm{DP}(H, \alpha)$. The process is as follows:

1. Imagine a stick of length 1.
2. For $k = 1, 2, \ldots, \infty$:
   - Draw an independent sample $\beta_k \sim \mathrm{Beta}(1, \alpha)$.
   - Define a weight $\pi_k = \beta_k \prod_{j=1}^{k-1}(1 - \beta_j)$. This corresponds to breaking off a proportion β_k of the *remaining* stick length.
   - Draw an independent sample $\theta_k \sim H$ from the base distribution. This is the location of the "atom" of probability.

The resulting distribution P is a discrete probability measure composed of an infinite number of weighted atoms:

$$P = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}$$

where δ_{θ_k} is a Dirac delta mass centered at θ_k.

The crucial insight is that if one first constructs the random measure P using this stick-breaking process and then draws a sequence of i.i.d. samples $(X_n)_n$ from P, the distribution of this sequence is identical to the one generated by the Chinese Restaurant Process.

6.6 Presentation of the laboratory

My internship lasted 8 weeks. I started on Monday, June 2. Here is a weekly summary of what I did:

1. Study of Count-min and Count-sketch data structures, study of the Zipf distribution. Numerical experiments.
2. In-depth study of Count Sketch and reading related articles. Attendance at Vivatech.
3. Reading related articles, implementation of an MLE for the Zipf case, and study of performance differences between Counting Bloom Filter and Count Min Sketch.
4. Research on optimizations, study of MLE, study of Bayesian probabilities, contacted a math researcher for information on the Dirichlet Process.
5. Study of Bayesian probabilities, study of MLE, testing optimizations and other estimators, experiments with MLE, discussions with M. Fuzy, Hot & Cold model study.
6. Writing and coding for the Hot & Cold case.

7. Testing the algorithm on the Hot & Cold case, study of a possible extension to Zipf.

8. Most of the writing for the internship report, work on the extension to Zipf.

I completed my internship at the Gaspard-Monge Computer Science Laboratory (LIGM), a joint research unit of Gustave Eiffel University. It is a medium-sized research lab with three supervisory institutions: Gustave Eiffel University and the CNRS as the primary supervisors, and the École des Ponts ParisTech (ENPC) as the secondary supervisor. Gustave Eiffel University was formed in 2020 from the merger of several institutions, including Université Paris-Est Marne-la-Vallée (UPEM) and ESIEE Paris.

The laboratory consists of six research teams and over 160 members, including nearly 100 permanent researchers and teacher-researchers. With few exceptions, research staff at LIGM are employed by one of the supervisory institutions: as researchers at the CNRS or ENPC, and as teacher-researchers at Gustave Eiffel University (which encompasses ESIEE Paris and the former UPEM).

I was personally part of the Algorithmique discrète et applications (ADA) team, but I mainly worked alongside my supervisor, M. Kucherov, who is a director of research from the CNRS (a permanent in the laboratory). Especially since there were no team seminars during the school holidays. On the other hand, I had the opportunity to interact with the Bases de données, automates, analyse d'algorithmes et modèles (BAAM) team by attending several of their seminars (Raphaël was in this team). I also spent a significant amount of time with Mr. Fuzy, in addition to my supervisor, since he had previously collaborated with Mr. Kucherov on topics related to my internship project. This allowed the three of us to discuss and brainstorm the issues I encountered during my experiments.

I also had the chance to speak with various researchers I met during my internship about their work and research topics, and was given a comprehensible ressource by M. Vandekerkhove (a math researcher) when I was struggling on the Dirichlet Process.

In addition, I was supported by Ms. Palescandolo, the laboratory's administrative manager. She helped me with the administrative processes and explained how the lab operates and how it is funded.

Spending such a long time thinking about a single topic was very different from what I was used to, whether in preparatory classes or at ENS. Moreover, realizing that the solution might lie in an area I was completely unfamiliar with—and hadn't yet explored—was quite unsettling. Several times, I felt like I had run out of ideas and had to sit down and reconsider which directions were still worth pursuing, especially since my internship was highly exploratory.

I really enjoyed this research experience: the thinking process—alone or in a group—and reading scientific articles. I was delighted to unexpectedly come across topics such as Bayesian probability, which I had been wanting to explore for some time. I also appreciated the mix between theoretical and experimental aspects.

6.7 Vivatech

Viva Technology (also known as VivaTech) is an annual technology and innovation trade fair held in Paris, France, at the Paris Expo Porte de Versailles. Created in 2016 by Publicis Groupe and Groupe Les Echos, the event brings together startups, established companies, investors, researchers, and the general public. It features exhibitions, conferences, and demonstrations in fields such as artificial intelligence, sustainable development, health technologies, and digital transformation. The event is organized over four days, the first three being reserved for professionals, and the last open to the public.

I took advantage of my stay in Paris to visit this fair with a friend. The experience offered a perspective on technology that emphasized innovation and societal impact over purely technical aspects. This complemented the "orientation" and "discovery" components of my internship by exposing me to emerging trends and applications, as well as a different view over technology.