# Optimized Learned Count-Min Sketch

**Kyosuke Nishishita** **Atsuki Sato** **Yusuke Matsui**
The University of Tokyo
{nishishita, a_sato, matsui}@hal.t.u-tokyo.ac.jp

## Abstract

Count-Min Sketch (CMS) is a memory-efficient data structure for estimating the frequency of elements in a multiset. Learned Count-Min Sketch (LCMS) enhances CMS with a machine learning model to reduce estimation error under the same memory usage, but suffers from slow construction due to empirical parameter tuning and lacks theoretical guarantees on intolerable error probability. We propose Optimized Learned Count-Min Sketch (OptLCMS), which partitions the input domain and assigns each partition to its own CMS instance, with CMS parameters $(\epsilon, \delta)$ analytically derived for fixed thresholds, and thresholds optimized via dynamic programming with approximate feasibility checks. This reduces the need for empirical validation, enabling faster construction while providing theoretical guarantees under these assumptions. OptLCMS also allows explicit control of the allowable error threshold, improving flexibility in practice. Experiments show that OptLCMS builds faster, achieves lower intolerable error probability, and matches the estimation accuracy of LCMS.

## 1 Introduction

Count-Min Sketch (CMS) is a probabilistic data structure that estimates the frequency of each element in a multiset in a memory-efficient way [1]. Frequency estimation is fundamental, and sketch data structures such as CMS are widely used [2, 3, 4, 5, 6, 7, 8]. While CMS is a useful data structure, there is a trade-off between estimation error and memory usage: reducing memory consumption increases the estimation error, while achieving lower error requires more memory.

Hsu et al. [9] proposed the Learned Count-Min Sketch (LCMS), which trains a machine learning model to generate approximate frequency scores, using a threshold to decide whether an element is stored in a dictionary or counted by CMS. Under a fixed memory budget, LCMS tunes CMS parameters and the threshold via validation data to minimize error, achieving more memory-efficient estimation than CMS. However, the construction of LCMS is slow because it requires validation. Also, LCMS lacks a theoretical guarantee on the probability of an "intolerable error," i.e., that the estimation error exceeds a user-specified threshold. Classical CMS provides such bounds, ensuring reliability, but LCMS does not.

We propose the Optimized Learned Count-Min Sketch (OptLCMS) to address these issues. Our method targets the Partitioned Learned Count-Min Sketch (PL-CMS) [10], a recently introduced data structure for the heavy hitter problem, and adapts it for frequency estimation. We optimize its parameters by partitioning the score space and formulating an optimization problem to minimize the probability of intolerable error. For fixed thresholds, $\delta$ can be derived analytically via the Karush-Kuhn-Tucker (KKT) conditions, while thresholds themselves are optimized through dynamic programming with approximate feasibility checks. CMS dimensions are obtained by relaxing integer constraints. This approach enables near-analytical construction, reduces build time compared to LCMS while maintaining memory efficiency. It also offers flexible error control with stronger guarantees.

ML For Systems workshop at Neural Information Processing Systems (NeurIPS 2025).

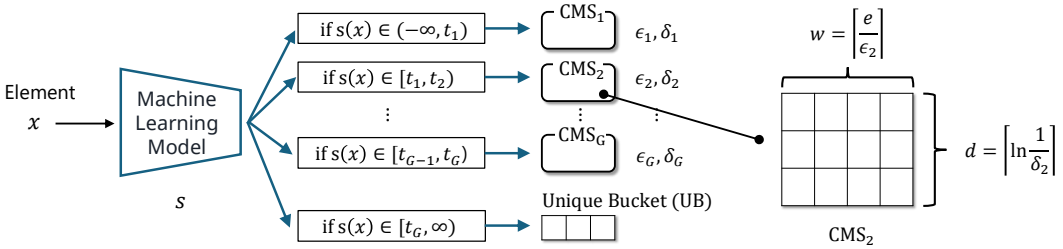


Figure 1: Partitioned Learned Count-Min Sketch

## 2 Related Work

Learned data structures were introduced by Kraska et al. [11] with "Learned Indexes," which reinterpret classical indexes such as B-Tree [12, 13] as models mapping keys to record positions. Subsequent work extended the idea from one-dimensional [14, 15, 16] to multi-dimensional ones [17, 18, 19]. Kraska et al. [11] also observed that probabilistic data structures, e.g., Bloom Filters [20], can be learned, inspiring Learned Bloom Filters [21, 22]. The Partitioned Learned Bloom Filter (PLBF) [22, 23] partitions the score space to optimize the false positive rate. Bloom Filters and Count-Min Sketch (CMS) share a space-efficient probabilistic design: the former tests set membership, the latter estimates frequencies. We extend the partitioning and optimization ideas from PLBF to CMS.

In the context of Learned Count-Min Sketch (LCMS), Hsu et al. [9] proposed combining CMS with a machine learning model, followed by Zhang et al. [24], which uses a threshold to route elements with low scores to CMS and high scores to the model's estimate. This approach depends heavily on model accuracy and cannot asymptotically eliminate error even with large memory. Nguyen et al. [10] introduced a partitioned LCMS for the heavy hitter problem, differing from frequency estimation. Their threshold is fixed a priori, whereas our method computes it via a dynamic programming approach to approximate the optimum.

## 3 Preliminaries

**Multiset.** A multiset is a set that allows multiple occurrences of the same element. For example, $\mathcal{A} = \{a, a, b\}$ and $\mathcal{B} = \{a, b, b\}$ are different, with $a$ occurring twice in $\mathcal{A}$ and once in $\mathcal{B}$. All sets in this paper are multisets.

**Count-Min Sketch (CMS).** The parameters of CMS are $\epsilon$ and $\delta$, where $0 < \epsilon$ and $0 < \delta \leq 1$. CMS estimates the frequency $f(x)$ of an element $x$ in a multiset using a table of width $w = \lceil e/\epsilon \rceil$ and depth $d = \lceil \ln(1/\delta) \rceil$ with $d$ hash functions. For each element in a multiset, $d$ hash functions specify $d$ cells. The value in each of these cells is incremented by one. The estimate $\hat{f}(x)$ is then given by the minimum of the $d$ cell values. CMS guarantees $\Pr[\hat{f}(x) - f(x) > \epsilon N] < \delta$ where $N$ is the size of the multiset, and requires $w \cdot d \cdot b$ bytes of memory, where $b$ is bytes per cell. Details of CMS are given in Appendix A.

**Unique Bucket (UB).** UB is an exact frequency counter, essentially a dictionary with keys given by the elements and values given by their exact counts, which requires significantly more memory than CMS.

## 4 Proposed Method

We propose the Optimized Learned Count-Min Sketch (OptLCMS), which extends the Partitioned Learned Count-Min Sketch (PL-CMS) [10]. While PL-CMS was developed for the heavy hitter problem, we adapt it to frequency estimation and formulate a optimization problem to determine its parameters. By partitioning the score space and analytically solving this problem, we minimize the probability that the estimation error exceeds a user-specified threshold under a fixed memory budget.

**Structure.** PL-CMS consists of a pre-trained machine learning model, $G$ CMS tables, and a unique bucket (UB) for high-frequency elements (Figure 1). Elements are routed based on the model score: high-score items go to UB for exact counting, others to the CMS of the corresponding score group. This design preserves the CMS property $\hat{f}(x) \geq f(x)$. We denote by $\boldsymbol{t} = [t_1, \ldots, t_G]^\top$ the vector of partition thresholds with $t_1 < t_2 < \cdots < t_G$, and let $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_G]^\top$ and $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_G]^\top$ be the CMS parameter vectors for the $G$ groups.

**Optimization formulation.** Our task is to efficiently determine optimal thresholds $\boldsymbol{t}$ and CMS parameters $\boldsymbol{\epsilon}, \boldsymbol{\delta}$, given the pre-trained ML model and the input multiset, under a memory budget. To this end, we formulate the following optimization problem

The upper bound on the intolerable error probability is $\sum_{g=1}^{G} \delta_g q_g$ which we minimize subject to the memory budget and CMS constraints:

$$\min_{\boldsymbol{t},\boldsymbol{\epsilon},\boldsymbol{\delta}} \quad \sum_{g=1}^{G} \delta_g q_g \tag{1}$$

$$\text{subject to} \quad b \sum_{g=1}^{G} \left\lceil \frac{e}{\epsilon_g} \right\rceil \left\lceil \ln \frac{1}{\delta_g} \right\rceil + cn = M \tag{2}$$

$$\delta_g \leq 1, \quad g = 1, \ldots, G \tag{3}$$

$$\epsilon_g = \frac{\epsilon N}{N_g}, \quad g = 1, \ldots, G \tag{4}$$

Here, $n$ denotes the number of unique elements classified into the UB, and $c$ is the memory usage per element. Thus, $cn$ corresponds to the total memory consumption (in bytes) of the UB. Equation (2) specifies the memory budget, where $M$ is the total available memory. Equation (3) enforces the CMS condition $\delta_g \leq 1$, and Equation (4) ensures that each CMS complies with the global allowable error. The ceiling functions enforce integer table dimensions in practice; however, for theoretical analysis, we relax them to continuous variables, introducing only a negligible non-convexity in implementation. The derivation of this formulation is in Appendix B.

**Analytical solution for $\boldsymbol{\delta}$.** With fixed thresholds $\boldsymbol{t}$, the problem is convex in $\boldsymbol{\delta}$ and can be solved analytically via the Karush-Kuhn-Tucker (KKT) conditions. The detailed derivation is provided in Appendix B. The closed-form solution for each $g \in \{1, \ldots, G\}$ is:

$$\delta_g = \min \left\{ 1, \ \frac{1}{q_g \epsilon_g} \exp \left[ - \frac{\frac{M - cn}{be} - I}{\sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g}} \right] \right\}, \tag{5}$$

where

$$I = \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln \left( q_g \epsilon_g \right), \quad \mathcal{D} = \{ g \mid \delta_g < 1 \}. \tag{6}$$

This gives $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$ for the $G$ CMSs when $\boldsymbol{t}$ is fixed.

**Threshold optimization.** Substituting $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$ into the objective yields an expression involving the Kullback–Leibler (KL) divergence between the data and query distributions of each group. The optimal thresholds are found by maximizing this divergence via dynamic programming (DP), ensuring $\delta_g < 1$ for all groups. The full derivation, DP recurrence, and proof of correctness are in Appendix B.

## 5 Experiments

We evaluate OptLCMS on the AOL query log (21M queries, 3.8M unique terms, Zipfian distribution) under two query patterns: *uniform* (each term once) and *frequency-weighted* (proportional to term frequency). All methods use the learned model of [9], with amortized model size $0.0152$ MB. This constant cost is excluded from Equation (2) but included in the reported memory usage for Figure 2 and 3. Implementation and dataset details are provided in Appendix C.

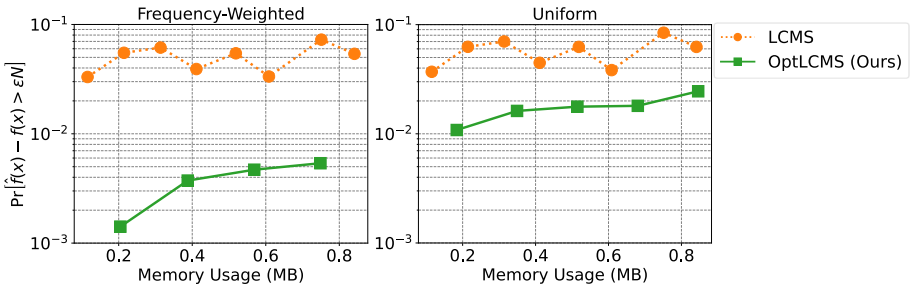


Figure 2: Memory Usage vs. intolerable error probability, defined as $\Pr\left[\hat{f}(x) - f(x) > \epsilon N\right]$. A lower position on the vertical axis indicates better performance.

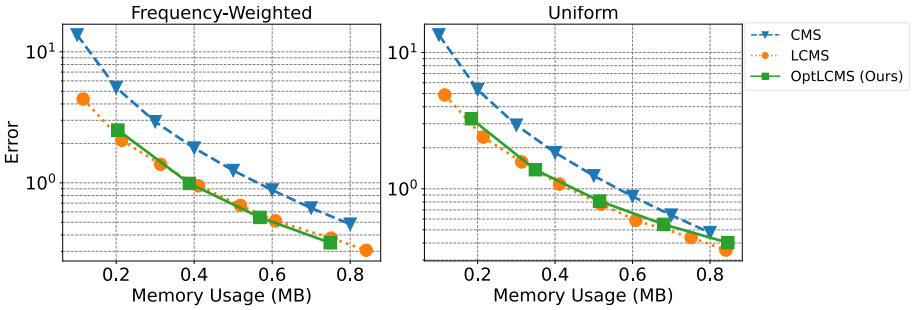


Figure 3: Memory Usage vs. Error. Curves closer to the bottom-left corner perform better.

| Data Structure | Construction [s] | distribution |
|---|---|---|
| LCMS | 10.712 | frequency-weighted |
| LCMS | 10.250 | uniform |
| OptLCMS (Ours) | 0.003 **(-10.709)** | frequency-weighted |
| OptLCMS (Ours) | 4.873 **(-5.377)** | uniform |

Table 1: Comparison of construction time

**Baselines.** We compare against: (i) CMS [1], tuned to minimize error for a fixed memory budget; (ii) LCMS [9], using the same model; and (iii) OptLCMS, with $G = 10$. For each $M$, $\epsilon$ is set to the smallest value feasible under CMS, namely $\epsilon = \frac{e}{M}$.

**Results.** Figure 2 shows that our OptLCMS consistently yields the lowest intolerable error probability while matching LCMS in average error, under both query patterns. OptLCMS achieves up to about 20× smaller intolerable error probability compared to LCMS. Figure 3 further confirms that minimizing the upper bound on intolerable error does not sacrifice average error performance.

Table 1 reports construction times. Unlike LCMS, which requires exhaustive threshold and parameter search with actual measurements, OptLCMS computes optimal parameters analytically from the validation data distribution, yielding orders-of-magnitude faster construction for frequency-weighted queries and more than twice as fast for uniform queries.

## 6 Conclusion and Future Work

We presented OptLCMS, a theoretically grounded variant of LCMS [9] that analytically determines CMS parameters by partitioning the score space, thereby minimizing the upper bound on the probability of intolerable error under a fixed memory budget. This approach removes the need for empirical validation, enabling much faster construction while maintaining comparable estimation accuracy to LCMS and providing stronger guarantees and flexibility. Due to time and scope constraints, we did not evaluate on the CAIDA dataset [9] or compare with confidence-aware LCMS [25]; extending our evaluation to these settings is left for future work.

## References


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

[2] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21(3): 270–313, 2003.

[3] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In *USENIX Conference on Hot Topics in Security*, pages 1–8. USENIX Association, 2010.

[4] Amit Goyal, Hal Daumé III, and Graham Cormode. Sketch algorithms for estimating point queries in nlp. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

[5] Partha Talukdar and William Cohen. Scaling graph-based semi supervised learning to large number of labels using count-min sketch. In *Artificial Intelligence and Statistics*, 2014.

[6] Fabon Dzogang, Thomas Lansdall-Welfare, Saatviga Sudhahar, and Nello Cristianini. Scalable preference learning from data streams. In *International Conference on World Wide Web*, pages 885–890, 2015.

[7] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. Netcache: Balancing key-value stores with fast in-network caching. In *Symposium on Operating Systems Principles*, pages 121–136, 2017.

[8] Amirali Aghazadeh, Ryan Spring, Daniel LeJeune, Gautam Dasarathy, Anshumali Shrivastava, et al. Mission: Ultra large-scale feature selection using count-sketches. In *International Conference on Machine Learning*, 2018.

[9] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019.

[10] Thuy Trang Nguyen and Cameron N Musco. Partitioned-learned count-min sketch. `https://openreview.net/forum?id=7W4boWjb3Q`, 2024.

[11] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *International Conference on Management of Data*, 2018.

[12] Rudolf Bayer and Edward McCreight. Organization and maintenance of large ordered indices. In *ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, pages 107–141, 1970.

[13] Douglas Comer. Ubiquitous b-tree. *ACM Computing Surveys*, 11(2):121–137, 1979.

[14] Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, David Lomet, and Tim Kraska. Alex: An updatable adaptive learned index. In *ACM SIGMOD International Conference on Management of Data*, pages 969–984, 2020.

[15] Paolo Ferragina and Giorgio Vinciguerra. The pgm-index: a fully-dynamic compressed learned index with provable worst-case bounds. *Proc. VLDB Endow.*, 13(8):1162–1175, 2020.

[16] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. Radixspline: a single-pass learned index. In *International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, pages 1–5, 2020.

[17] Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. Learning multi-dimensional indexes. In *ACM SIGMOD International Conference on Management of Data*, pages 985–1000, 2020.

[18] Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. Tsunami: a learned multi-dimensional index for correlated data and skewed workloads. *Proc. VLDB Endow.*, 14(2): 74–86, 2020.

[19] Fuma Hidaka and Yusuke Matsui. Flexflood: Efficiently updatable learned multi-dimensional index. In *Machine Learning for Systems Workshop, NeurIPS 2024*, 2024.

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

[21] Zhenwei Dai and Anshumali Shrivastava. Adaptive learned bloom filter (ada-bf): Efficient utilization of the classifier with application to real-time information filtering on the web. In *Advances in Neural Information Processing Systems*, pages 11700–11710, 2020.

[22] Kapil Vaidya, Eric Knorr, Michael Mitzenmacher, and Tim Kraska. Partitioned learned bloom filters. In *International Conference on Learning Representations*, 2021.

[23] Atsuki Sato and Yusuke Matsui. Fast partitioned learned bloom filter. *Advances in Neural Information Processing Systems*, 36:39119–39146, 2023.

[24] Meifan Zhang, Hongzhi Wang, Jianzhong Li, and Hong Gao. Learned sketches for frequency estimation. *Information Sciences*, 507:365–385, 2020.

[25] Anders Aamand, Justin Chen, Huy Nguyen, Sandeep Silwal, and Ali Vakilian. Improved frequency estimation algorithms with and without predictions. *Advances in Neural Information Processing Systems*, 36:14387–14399, 2023.

[26] Lada A Adamic and Bernardo A Huberman. Zipf's law and the internet. *Glottometrics*, 3(1): 143–150, 2002.

[27] Steven T Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, 21:1112–1130, 2014.

# A Count-Min Sketch (CMS) Details

Count-Min Sketch (CMS) is a probabilistic data structure for estimating the frequency $f(x)$ of an element $x$ in a multiset $\mathcal{U}$. CMS maintains a two-dimensional array $C$ of width

$$w \coloneqq \left\lceil \frac{e}{\epsilon} \right\rceil \tag{7}$$

and depth

$$d \coloneqq \left\lceil \ln \frac{1}{\delta} \right\rceil, \tag{8}$$

where $\epsilon > 0$ and $0 < \delta \leq 1$ are user-specified parameters controlling accuracy and confidence, respectively.

### A.1 Hash Functions

We prepare $d$ pairwise-independent hash functions

$$h_i : \mathcal{S} \to \{0, 1, \ldots, w-1\}, \quad i = 0, 1, \ldots, d-1, \tag{9}$$

where $\mathcal{S}$ denotes the set of all unique elements that can appear in $\mathcal{U}$.

### A.2 Update Procedure

For each occurrence of an element $x$, the count in each row is incremented as:

$$C[i,\, h_i(x)] \;\leftarrow\; C[i,\, h_i(x)] + 1, \quad \forall i \in \{0, \ldots, d-1\}. \tag{10}$$

### A.3 Query Procedure

The estimated frequency $\hat{f}(x)$ is given by:

$$\hat{f}(x) \coloneqq \min_{i \in \{0,\ldots,d-1\}} C[i,\, h_i(x)], \tag{11}$$

which mitigates the effect of overcounting caused by hash collisions.

### A.4 Error Guarantee

By construction, CMS guarantees $\hat{f}(x) \geq f(x)$ and satisfies the probabilistic bound:

$$\Pr\left[\hat{f}(x) - f(x) > \epsilon N\right] < \delta, \tag{12}$$

where $N \coloneqq |\mathcal{U}|$ is the total number of elements in the multiset.

### A.5 Memory Usage

The total memory consumption is:

$$\text{Memory} = \left\lceil \frac{e}{\epsilon} \right\rceil \cdot \left\lceil \ln \frac{1}{\delta} \right\rceil \cdot b \quad [\text{bytes}], \tag{13}$$

where $b$ is the memory usage per counter (in bytes).

# B Derivation of the Optimization Problem

### B.1 Objective: Probability Upper Bound to Minimize

Our goal is to minimize the probability that the estimation error of the LCMS exceeds the allowable error $\epsilon N$. We refer to this quantity as the *intolerable error probability*. To analyze this, we partition the score space from the learned model into $G+1$ groups, where $G$ is a hyperparameter, and associate a Count-Min Sketch (CMS) with each group.

Let the threshold vector be $\boldsymbol{t} = [t_1, \ldots, t_G]^\top$, where $t_1 < t_2 < \cdots < t_G$. For the $g$-th group, we define the CMS parameters $\epsilon_g$ and $\delta_g$, and denote the sets handled by each CMS as $\mathcal{U}_g$ with size $N_g = |\mathcal{U}_g|$. We also define the query distribution $q_g = \Pr[x \in \mathcal{U}_g]$. The parameter condition

$$\epsilon_g N_g = \epsilon N \tag{14}$$

is imposed so that the allowable error for each CMS matches that of the entire system.

We consider the probability that the estimation error exceeds $\epsilon N$:

$$\Pr\left[\hat{f}(x) - f(x) > \epsilon N\right] = \sum_{g=1}^{G} \Pr\left[\hat{f}(x) - f(x) > \epsilon N \mid x \in \mathcal{U}_g\right] \Pr[x \in \mathcal{U}_g] \tag{15}$$

$$< \sum_{g=1}^{G} \delta_g q_g, \tag{16}$$

where the last inequality follows from applying the CMS error bound to each group. Thus, minimizing the intolerable error probability reduces to minimizing $\sum_{g=1}^{G} \delta_g q_g$ over the partitioning thresholds $\boldsymbol{t}$ and CMS parameters $(\epsilon_g, \delta_g)$, subject to the overall memory constraint.

### B.2 Problem Formulation

The optimization problem is:

$$\min_{\boldsymbol{t}, \boldsymbol{\epsilon}, \boldsymbol{\delta}} \quad \sum_{g=1}^{G} \delta_g q_g \tag{17}$$

$$\text{s.t.} \quad b \sum_{g=1}^{G} \left\lceil \frac{e}{\epsilon_g} \right\rceil \left\lceil \ln \frac{1}{\delta_g} \right\rceil + cn = M \tag{18}$$

$$\delta_g \leq 1, \quad g = 1, \ldots, G \tag{19}$$

$$\epsilon_g = \frac{\epsilon N}{N_g}, \quad g = 1, \ldots, G \tag{20}$$

where $n$ is the number of unique elements in the UB, $c$ the per-element memory, and $M$ the total budget.

### B.3 Solution of the optimization problem

**Lagrangian Function** We now apply the Karush–Kuhn–Tucker (KKT) conditions to derive the optimal solution. This involves setting the derivatives of the Lagrangian with respect to each variable to zero and applying the complementary slackness condition. The Lagrangian function for the problem is given by:

$$\mathcal{L}(\boldsymbol{\delta}, \boldsymbol{\mu}, \lambda) = \sum_{g=1}^{G} \delta_g q_g + \sum_{g=1}^{G} \mu_g \left(\delta_g - 1\right) + \lambda \left( b \sum_{g=1}^{G} \frac{e}{\epsilon_g} \ln \frac{1}{\delta_g} + cn - M \right). \tag{21}$$

**First-Order Conditions** Taking the partial derivatives of the Lagrangian with respect to $\delta_g$, $\mu_g$, and $\lambda$, we obtain the following conditions.

The derivative with respect to $\delta_g$ is:

$$\frac{\partial \mathcal{L}}{\partial \delta_g} = q_g + \mu_g - \lambda b \frac{e}{\epsilon_g} \frac{1}{\delta_g}. \tag{22}$$

Setting this equal to zero for the optimal condition:

$$q_g + \mu_g - \lambda b \frac{e}{\epsilon_g} \frac{1}{\delta_g} = 0. \tag{23}$$

The derivative with respect to $\lambda$ is:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = b \sum_{g=1}^{G} \frac{e}{\epsilon_g} \ln \frac{1}{\delta_g} + cn - M. \tag{24}$$

Setting this equal to zero for the optimal condition:

$$b \sum_{g=1}^{G} \frac{e}{\epsilon_g} \ln \frac{1}{\delta_g} + cn - M = 0. \tag{25}$$

The derivative with respect to $\mu_g$ is:

$$\frac{\partial \mathcal{L}}{\partial \mu_g} = \delta_g - 1. \tag{26}$$

**KKT Conditions** From the complementary slackness condition, we obtain the following:

$$\mu_g(\delta_g - 1) = 0. \tag{27}$$

This implies that either $\mu_g = 0$ or $\delta_g = 1$, meaning that the constraint is either inactive or active. From the non-negativity of the Lagrange multiplier, we have $\mu_g \geq 0$. From the non-negativity of the constraint, it follows that $\delta_g \leq 1$.

Consider the case $\mu_g = 0$. When $\mu_g = 0$, from the complementary slackness condition, we have $\delta_g < 1$. The optimization condition (23) becomes:

$$q_g - \lambda b \frac{e}{\epsilon_g} \frac{1}{\delta_g} = 0, \tag{28}$$

which can be solved for $\delta_g$ as:

$$\delta_g = \frac{\lambda b e}{q_g \epsilon_g}. \tag{29}$$

We then check whether this $\delta_g$ satisfies the condition $\delta_g \leq 1$. If $\delta_g \geq 1$, this case is not valid.

Consider the case $\delta_g = 1$. When $\delta_g = 1$, the complementary slackness condition implies that $\mu_g \geq 0$. The optimization condition becomes:

$$q_g + \mu_g - \lambda b \frac{e}{\epsilon_g} = 0, \tag{30}$$

which gives:

$$\mu_g = \lambda b \frac{e}{\epsilon_g} - q_g. \tag{31}$$

**Derivation of $\delta_g$** Let $\mathcal{D}$ be the set of $g \in \{1, \ldots, G\}$ such that $\delta_g < 1$, i.e.,

$$\mathcal{D} = \{g \mid \delta_g < 1\}. \tag{32}$$

In this case, for $g \in \mathcal{D}$, $\delta_g = \frac{\lambda b e}{q_g \epsilon_g}$, and for $g \notin \mathcal{D}$, $\delta_g = 1$.

Substituting $\delta_g$ into Equation (25), we get:

$$b \sum_{g \in \mathcal{D}} \frac{e}{\epsilon_g} \ln \frac{1}{\frac{\lambda b e}{q_g \epsilon_g}} + cn - M = 0 \tag{33}$$

$$b \sum_{g \in \mathcal{D}} \frac{e}{\epsilon_g} \ln \frac{q_g \epsilon_g}{\lambda b e} + cn - M = 0 \tag{34}$$

$$\sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln (q_g \epsilon_g) - \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln (\lambda b e) = \frac{M - cn}{be} \tag{35}$$

$$- \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln (\lambda b e) = \frac{M - cn}{be} - \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln (q_g \epsilon_g) \tag{36}$$

$$\lambda b e = \exp \left[ - \frac{\frac{M - cn}{be} - \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln (q_g \epsilon_g)}{\sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g}} \right] \tag{37}$$

Therefore, for $g \in \mathcal{D}$, $\delta_g$ is given by the following:

$$\delta_g = \frac{1}{q_g \epsilon_g} \exp\left[ -\frac{\frac{M-cn}{be} - \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln\left(q_g \epsilon_g\right)}{\sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g}} \right]. \tag{38}$$

From this, $\delta_g$ is expressed as follows:

$$\delta_g = \min\left\{ 1, \quad \frac{1}{q_g \epsilon_g} \exp\left[ -\frac{\frac{M-cn}{be} - \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \ln\left(q_g \epsilon_g\right)}{\sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g}} \right] \right\}. \tag{39}$$

### B.4 Threshold Determination Method

Given a threshold vector $\boldsymbol{t} = [t_1, \ldots, t_G]^\top$, we can analytically derive the CMS parameters as the solution to the optimization problem in Equation (17), as shown in Equation (5). Thus, the next task is to find the optimal threshold vector $\boldsymbol{t} = [t_1, \ldots, t_G]^\top$.

To determine the threshold, we substitute $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$ into the objective function, which we aim to minimize. We rewrite the objective function (17) as follows:

$$\sum_{g \in \overline{\mathcal{D}}} q_g + \sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g} \exp\left[ -\frac{\epsilon N (M - cn)}{be \sum_{g \in \mathcal{D}} N_g} \right] \exp\left[ \frac{I}{\sum_{g \in \mathcal{D}} \frac{1}{\epsilon_g}} \right], \tag{40}$$

where $\overline{D} = \{g \mid \delta_g = 1\}$.

Considering that the task to solve is element frequency estimation, we impose the condition $\delta_g \neq 1$. This is because $\delta_g = 1$ implies that the CMS table for group $g$ does not exist. While $\delta_g = 1$ may be a valid result for minimizing the probability that the estimation error exceeds the allowable error, a table is necessary to count the element frequencies. Therefore, we consider $\delta_g < 1$ going forward.

In this case, the objective function to be minimized is as follows:

$$\sum_{g=1}^{G} \frac{1}{\epsilon_g} \exp\left[ -\frac{\epsilon N (M - cn)}{be \sum_{g=1}^{G} N_g} \right] \exp\left[ \frac{I}{\sum_{g=1}^{G} \frac{1}{\epsilon_g}} \right]. \tag{41}$$

Let the size of the multiset consisting of the elements classified into the UB be denoted as $N_{\mathrm{UB}}$. Note that $n$ refers to the size of the set of elements classified into the UB, while $N_{\mathrm{UB}}$ represents the size of the multiset, e.g. $n = 2$ and $N_{\mathrm{UB}} = 3$ for $\{x_1, x_2, x_2\}$. Additionaly, for each $g \in \{1, \ldots, G\}$ we define new variables as follows:

$$u_g = \frac{N_g}{N - N_{\mathrm{UB}}}, \tag{42}$$

$$v_g = \frac{q_g}{1 - q_{\mathrm{UB}}}. \tag{43}$$

$u_g$ represents the size of the multiset handled by the CMS for group $g$ relative to the size of the multiset handled by the CMS, and it satisfies $\sum_{g=1}^{G} v_g = 1$. Additionally, $v_g$ represents the proportion of queries handled by the CMS for group $g$ out of all queries processed by the CMS, and it satisfies $\sum_{g=1}^{G} v_g = 1$. Substituting these into $\epsilon_g$ and $q_g$, we obtain the following:

$$\epsilon_g = \frac{\epsilon N}{(N - N_{\mathrm{UB}}) u_g}, \tag{44}$$

$$q_g = v_g (1 - q_{\mathrm{UB}}). \tag{45}$$

Substituting these into Equation (41) and simplifying, the objective function to minimize becomes:

$$(1 - q_{\mathrm{UB}}) \exp\left[ -\frac{\epsilon N (M - cn)}{be(N - N_{\mathrm{UB}})} \right] \exp\left[ -\sum_{g=1}^{G} u_g \ln \frac{u_g}{v_g} \right]. \tag{46}$$

By determining the threshold $t_G$ that defines the boundary between the UB and the CMS, we can automatically determine the variables $q_{\mathrm{UB}}, n$, and $N_{\mathrm{UB}}$. Therefore, what Equation (46) suggests is that by setting $t_G$, the partitioning of the score space that is smaller than $t_G$ should be done in such a way that $\sum_{g=1}^{G} u_g \ln \frac{u_g}{v_g}$ is maximized. The term $\sum_{g=1}^{G} u_g \ln \frac{u_g}{v_g}$ represents the Kullback-Leibler (KL) divergence.

### B.5 Derivation of the Threshold

This maximization problem of the KL divergence can be efficiently solved using dynamic programming (DP) as follows.

$$DP(s,p) = \max_{y} \left\{ DP(y,p-1) + u(y,s) \ln \frac{u(y,s)}{v(y,s)} \right\}. \tag{47}$$

Here, $DP(s,p)$ represents the maximum divergence when the score space with scores less than $s$ is divided into no more than $p$ parts. Additionally, $u(y,s)$ represents the proportion of the size of the multiset consisting of elements with scores in the interval $[y,s)$, relative to the size of the multiset processed by the CMS. Meanwhile, $v(y,s)$ represents the proportion of queries processed by the CMS that correspond to elements with scores in the interval $[y,s)$.

The following explains the $DP(s,p)$ process. First, prepare a list to store the sorted threshold candidates in advance.

1. Assign $DP(s,p-1)$ to $DP(s,p)$.
2. Iterate through the threshold candidates that are less than $s$. Let this be denoted as $y$.
3. Consider dividing the score space using the threshold $y$, and compute $DP(y,p-1) + u(y,s) \ln \frac{u(y,s)}{v(y,s)}$.
4. If $DP(y,p) < DP(y,p-1) + u(y,s) \ln \frac{u(y,s)}{v(y,s)}$, verify whether the CMS handling the score space $[y,s)$ (which is denoted as group $g$) satisfies the condition for $\delta_g$ .
5. If the condition is satisfied, update $DP(y,p)$ as $DP(y,p) = DP(y,p-1) + u(y,s) \ln \frac{u(y,s)}{v(y,s)}$.

This DP solution would return an analytical optimal solution if the condition $\delta_g < 1$ were not present, as the process in step Item 4 would not be required. However, since CMS requires the condition $\delta_g < 1$, the process in step Item 4 is necessary.

Next, we describe how to check whether each $\delta_g$ satisfies the condition $\delta_g < 1$ when adopting the new threshold $y$. Now, we are considering an efficient solution using DP, so the division of the score space $[s,t_G)$ is unknown. Therefore, at this stage, it is impossible to determine $\delta_g$ precisely. Therefore, we approximate $\delta_g$ as $\hat{\delta}_g$ as follows:

$$\hat{\delta}_g = \frac{u_g}{v_g} \exp\left[ -\frac{\epsilon N(M-cn)}{be(N-N_{\mathrm{UB}})} \right] \exp\left[ -\left( DP(s,p) + u(s,t_G) \ln \frac{u(s,t_G)}{v(s,t_G)} \right) \right]. \tag{48}$$

If $\hat{\delta}_g < 1 \implies \delta_g < 1$ holds, then regardless of how the unscanned score space is divided, we only need to check that $\hat{\delta}_g < 1$. To demonstrate that $\hat{\delta}_g < 1 \implies \delta_g < 1$, we derive the following lemma.

**Lemma 1.** *For any positive numbers $j, j_1, j_2, k, k_1, k_2 > 0$, the following inequality holds:*

$$j \ln\left(\frac{j}{k}\right) \le j_1 \ln\left(\frac{j_1}{k_1}\right) + j_2 \ln\left(\frac{j_2}{k_2}\right),$$

*where $j = j_1 + j_2$ and $k = k_1 + k_2$*

*Proof.* By using Jensen's inequality, since the logarithmic function $\ln(x)$ is concave, the following inequality holds:

$$\ln\left(w_1 x_1 + w_2 x_2\right) \ge w_1 \ln(x_1) + w_2 \ln(x_2). \tag{49}$$

Let $w_1 = \frac{j_1}{j_1+j_2}$, $w_2 = \frac{j_2}{j_1+j_2}$, $x_1 = \frac{k_1}{j_1}$, and $x_2 = \frac{k_2}{j_2}$, then

$$\ln\left(\frac{k_1+k_2}{j_1+j_2}\right) \ge \frac{j_1}{j_1+j_2} \ln\left(\frac{k_1}{j_1}\right) + \frac{j_2}{j_1+j_2} \ln\left(\frac{k_2}{j_2}\right).$$

By multiplying both sides by $j_1 + j_2$ and applying the logarithmic property of division, we obtain

$$(j_1 + j_2) \ln \left( \frac{j_1 + j_2}{k_1 + k_2} \right) \leq j_1 \ln \left( \frac{j_1}{k_1} \right) + j_2 \ln \left( \frac{j_2}{k_2} \right).$$

This is the inequality to be proven, which is equal to

$$j \ln \left( \frac{j}{k} \right) \leq j_1 \ln \left( \frac{j_1}{k_1} \right) + j_2 \ln \left( \frac{j_2}{k_2} \right).$$

□

**Proposition 1.**

$$\hat{\delta}_g < 1 \implies \delta_g < 1.$$

*Proof.* From Lemma 1, the following inequality holds:

$$u(s, t_G) \ln \frac{u(s, t_G)}{v(s, t_G)} \leq \sum_{g=p}^{G} u_g \ln \frac{u_g}{v_g}.$$

Therefore, when the interval $[s, t_G)$ is divided, the KL divergence increases compared to the case when no division is made. Thus, we have:

$$\hat{\delta}_g < 1 \implies \delta_g < 1.$$

□

What this proposition shows is that if $\hat{\delta_g} < 1$ holds at the current stage,then $\delta_g < 1$ will hold regardless of how the remaining unexamined score space is divided. Therefore, the new threshold can be determined as the value of $y$ that maximizes $DP(s, p)$, when the interval $[y, s)$ is treated as a new group, and the corresponding $\hat{\delta}_g$ for this group satisfies $\hat{\delta}_g < 1$. This allows for the efficient approximation of the optimal threshold.

# C Experimental Setup

## C.1 Dataset

We use the **AOL query log dataset**, which contains 21 million search queries collected from approximately 650,000 users over 90 days.

- **Unique terms:** 3.8 million
- **Examples:** "google", "yahoo", "amazon.com"
- **Distribution:** Follows *Zipf's law*, which is also observed in web traffic patterns [26] and natural language word frequencies [27].
- **Data split:** Queries from the 5th day are used for training/parameter tuning, and queries from the 50th day are used for evaluation/counting.

## C.2 Query Distributions

We evaluate two types of query distributions:

1. **Uniform Distribution:** Each unique term is queried exactly once.
2. **Frequency-Weighted Distribution:** Each unique term is queried according to its observed frequency in the dataset.

### C.3 Implementation Details

- **Unique Bucket (UB):** Implemented as an open-addressing hash table without bucket expansion.
- **Per-element UB memory cost:** $c = 20$ bytes.
- **Learned model:** We adopt the model from Hsu et al. [9], trained on the first 5 days of AOL query logs.
- **Model amortized memory usage:** $0.0152$ MB over 90 days (following Hsu et al. [9]).
- **Hardware:** Intel(R) Core(TM) Ultra 7 155H CPU, 16 cores, 3.80 GHz.

### C.4 Comparison Methods

We compare the following methods under the same total memory budget:

- **Count-Min Sketch (CMS)** [1]:
  - Hyperparameters: $\epsilon, \delta$.
  - Under a fixed memory budget, parameters that minimize estimation error are non-trivial to derive.
  - We empirically search multiple parameter settings and report the best-performing configuration.
- **Learned Count-Min Sketch (LCMS)** [9]:
  - Hyperparameter: memory usage.
- **Optimized LCMS (OptLCMS)** (proposed):
  - Hyperparameters: memory usage, allowable error $\epsilon$, and maximum number of partitions $G$.
  - $\epsilon$ is set to the smallest value achievable by CMS under the same memory ($\epsilon = e/M$).
  - $G = 10$ in all experiments.