Fair-Count-Min: Frequency Estimation under Equal Group-wise Approximation Factor

Anonymous Author(s)

Abstract

Frequency estimation in streaming data often relies on sketches like Count-Min to provide approximate answers with sublinear space. However, Count-Min sketches introduce additive errors that disproportionately impact the unpopular elements, creating fairness concerns across different groups of elements. We introduce Fair-Count-Min, a frequency estimation sketch that guarantees equal expected approximation factors across element groups, thus addressing the unfairness issue. We propose a column partitioning approach with group-aware semi-uniform hashing to eliminate collisions between elements from different groups. We provide theoretical guarantees for fairness, analyze its associated cost, and validate our findings through extensive experiments on real-world datasets in comparison with representative state-of-the-art baselines. Our experimental results demonstrate that Fair-Count-Min achieves fairness with generally small additional error while maintaining efficiency comparable to that of the Count-Min sketch.

1 Introduction

Estimating item frequencies in data streams is a fundamental problem in computer science, where given a universe of elements, the objective is to count the appearance of each within the stream. However, due to memory limitations and processing time in streaming settings, exact counting is often impractical. Consequently, approximating the counts using techniques such as the Count-Min (CM) sketch has become widely adopted [27]. The CM estimates the frequency of elements by randomly hashing the elements into a smaller set of buckets, while adding up the frequencies of all elements in each bucket. The frequency count of each bucket is then returned as an upper bound of the frequency of elements it contains.

The CM sketches offer space-efficient frequency estimations with small error guarantees. Nevertheless, *the error guarantees are additive*. Such errors may be negligible for popular elements with a high frequency in the data stream. However, as further elaborated in Example 2, the unpopular groups with low frequencies in the stream are disproportionately impacted under such guarantees. That is because even small additive errors can cause a significant relative increase in the count estimation of unpopular (low-frequency) elements. This imbalance gives rise to a fairness issue – an aspect overlooked in the existing work. Let us further illustrate the implications of this issue by presenting the following example:

Example 1: *Urban traffic sensors log vehicle IDs or license plate hashes as cars pass intersections. Buses, taxis, and delivery trucks show up hundreds of times per day, while private vehicles may only appear a few times. Suppose on average the public transportation cars pass the intersections 100 times, while this number is 5 for the private vehicles. In this setting, an additive error of 10 is tolerable for public transportation cars, but it increases the expected estimation for private vehicles by 200%. This distortion matters: traffic management systems could mistakenly flag private vehicles as suspicious, or misallocate toll discounts and congestion pricing.*[1]

To address this issue, in this paper, we introduce **Fair-Count-Min** (FCM), a frequency estimation sketch with the equal group-level approximation factor guarantee, ensuring that the relative count-estimation increase is equal for various groups, irrespective of their popularity in the data stream. To the best of our knowledge, FCM is the first frequency estimation sketch with provable *multiplicative error* guarantees in its estimation.

Our key idea in the design of the FCM is the partitioning of the sketch buckets strategy based on a group-aware, semi-uniform hashing scheme that prevents collisions between elements of different groups. Then, we prove that, carefully selecting the number of buckets allocated to each group, our FCM sketch guarantees an equal expected approximation factor across the groups, i.e., equal ratios of the true frequency to the estimated one. The FCM sketch applies to binary and non-binary grouping of the elements, while the grouping can be based on element popularity or any other grouping strategy, e.g., based on the demographic groups.

Since *our approach is restricted to the same memory capacity as CM*, FCM does not introduce any additional memory overhead. Furthermore, FCM and CM have the same time complexities for update and query operations. In other words, the time taken to look up the frequency estimation of an element and the time to increase the frequency of an element in the data stream are the same in CM and FCM.

In a general setting, where multiple hash functions are used in the FCM, identifying the proper number of buckets allocated to each group requires solving an equation, for which we propose efficient exact and approximation algorithms.

Furthermore, we theoretically analyze the price of fairness (PoF) of FCM. In general, PoF denotes the accuracy loss incurred when fairness constraints are imposed. It arises because accuracy-maximizing outcomes often disproportionately benefit certain groups, and enforcing fairness necessitates deviations from this optimum. Interestingly, we show that in this case achieving fairness may even reduce the total additive error for a specific case when the CM sketch uses a single random hash function, while experimentally demonstrating that the PoF is small for the general settings.

Beyond the theoretical analysis, we conduct extensive experiments on multiple real-world datasets, replicating our motivating examples to evaluate the practical performance of the proposed approach. In summary, our experiments verify (a) while CM is usually not fair, FCM achieves fairness in all experiments, (b) the PoF is generally small, and (c) FCM has the same memory and time efficiency as CM.

Summary of contributions:

- We propose a novel notion of *group fairness* in the frequency estimation context, defined by the requirement that the approximation factor (the *multiplicative* overestimation error) remains

[1] Additional motivation examples are provided in Appendix A.

Notation	Description
$\mathcal{D}$	data stream
$\mathcal{U}$	A universe of n element types
e	an element type
$f(.)$	true frequency of an element type
$\hat{f}(.)$	estimated frequency of an element type
$\mathcal{G}$	The set of ℓ groups
$\mathbf{g}$	a group with $n_{\mathbf{g}}$ element types
CM	count-min 2D array
w, d	width and depth of the sketch
$w_{\mathbf{g}}, d_{\mathbf{g}}$	# of columns and rows assigned to group $\mathbf{g}$
N	length of stream
$N_{\mathbf{g}}$	frequency of group $\mathbf{g}$
$\mathsf{h}(.)$	random hash function
$\varepsilon_A(.)$	additive error of an element type
$\alpha(.)$	approximation factor of an element type
C_i	total frequency in i-th bucket
PoF	price of fairness
$\mathcal{L}_{FCM}, \mathcal{L}_{CM}$	total additive errors of FCM and CM

Table 1: Table of notations

equal across different groups. This contrasts with traditional approaches based on *additive* error bounds, which tend to disproportionately impact unpopular groups. Using this notion of fairness, we introduce *Fair-Count-Min* (FCM), the frequency estimation sketch that guarantees group fairness.

- We introduce a *column-partitioning* technique based on semi-uniform hash functions to develop FCM. This method assigns groups to disjoint sets of hash buckets, thereby eliminating inter-group collisions and promoting fairness in estimation. Our proposed method, FCM, is agnostic to both the grouping strategy and the underlying distribution of the data elements. Furthermore, FCM does not increase the memory and time requirements of CM.
- We theoretically prove that FCM is fair, i.e., it ensures an equal expected approximation factor across groups. Furthermore, we analyze the *price of fairness* and show that it is typically small—and in some cases, can even be negative.
- We design both exact and approximate algorithms to efficiently compute the optimal allocation of buckets per group.
- We empirically evaluate FCM on several real-world datasets and compare it with representative state-of-the-art baselines for frequency estimation such as learning-based approach [48] and CM with conservative update [36]. The results show that FCM is the only approach that successfully achieves group fairness while incurring only a generally small additional additive error, all while preserving the time and space efficiency characteristics of CM. Interestingly, CM with conservative update–though effective in minimizing additive error–exhibits the poorest performance with respect to multiplicative error. The learned approach is constrained by its capacity on identifying heavy hitters from the data sample and incurs construction and query times several orders of magnitude higher than non-learned methods.

2 Preliminary

2.1 Data Model

Let $\mathcal{D}$ be a data stream consisting of elements (or events), each belonging to a type from a finite universe $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$. Each element type e be associated with a group $\mathbf{g} \in \mathcal{G}$, where $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_\ell\}$. Let N be the number of elements in the data stream $\mathcal{D}$. We use the function $f : \mathcal{U} \to [N]$ to refer to the frequency of each element type in $\mathcal{D}$. That is, $f(e)$ is the number of times an element of type e appears in $\mathcal{D}$. Formally,

$$f(e) = \sum_{x \in \mathcal{D}} \mathbb{1}(x = e),$$

where $\mathbb{1}(x = e)$ is the indicator function that equals 1 if x is of type e and 0 otherwise.

2.2 Count-Min Sketch

Frequency estimation is a fundamental problem in streaming data processing. Given a data stream $\mathcal{D}$, the goal is to efficiently estimate the frequency $f(e)$ of each element type e. It is easy to see that finding an exact solution to the above problem requires $\Theta(n)$ space, as it involves maintaining n counters for each element type. To overcome this limitation, frequency estimation data structures (a.k.a sketches) have been designed to provide approximate answers to such queries while using sub-linear space. In addition to being space-efficient, these sketches provide probabilistic error guarantees through tunable parameters and enable constant-time update and query operations.

CM is among the most widely used sketches for frequency estimation. It represents the frequency estimates using a two-dimensional array CM of d rows and w columns. The sketch employs d independent hash functions $\mathsf{h}_1, \mathsf{h}_2, \ldots, \mathsf{h}_d$, where each $\mathsf{h}_i : \mathcal{U} \to [w]$ maps an element type to a column index (called *bucket* or *bin* in the rest of the paper) in row i. These hash functions distribute element types across different buckets to reduce the impact of hash collisions. When a new element of type e arrives in the stream, the sketch updates its counts as follows:

$$CM[i, \mathsf{h}_i(e)] \leftarrow CM[i, \mathsf{h}_i(e)] + 1, \quad \forall i \in [d].$$

Each row records a count for e, though potential overestimations may occur due to hash collisions. To estimate the frequency of an element type[2] e, the sketch returns:

$$\hat{f}(e) = \min_{i=1}^{d} CM[i, \mathsf{h}_i(e)].$$

Taking the minimum across multiple rows mitigates over-counting errors. As previously mentioned, the estimation error in CM occurs due to hash collisions. For each element type e the estimated frequency $\hat{f}(e)$ is always an upper bound on the true frequency $f(e)$:

$$\hat{f}(e) = f(e) + \varepsilon_A(e), \tag{1}$$

where the additive error $\varepsilon_A(e)$ corresponds to the combined frequency of all other elements that hash into the same bucket as e. In expectation, the additive error $\varepsilon_A(e)$ is bounded by $\frac{N}{w}$. Formally:

$$\mathbb{E}[\hat{f}(e)] = f(e) + \frac{\sum_{\forall e' \neq e,\, \mathsf{h}(e') = \mathsf{h}(e)} f(e')}{w} \leq f(e) + \frac{N}{w} \tag{2}$$

Several variations of the CM exist, including the CM with conservative updates [36], the Count-Mean-Min sketch [30], and other related frequency estimation structures such as the Count sketch [21]

[2] In the rest of the paper, we use "element e" and "element type e" interchangeability to refer to elements of type e.

and Spectral Bloom Filters [25]. While using CM as the foundation for the development of our sketch, and deriving our theoretical analysis, *we compare our work against other baselines in experiments.*

2.3 Fairness: Equal Expected Approximation Factor

Existing CM sketches provide an additive upper-bound error ε_A on the frequency estimation of the elements. However, while ε_A may be a negligible error for the popular frequent elements, it can be intolerable for the unpopular ones which appear infrequently in the sequence. In other words, the significance of the frequency estimation error is relative to the frequency value. To further clarify this using a toy example, let us consider Example 2.

Example 2: *Consider a universe of events of types $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$, where $\{e_1, \cdots, e_{\frac{n}{2}}\}$ belong to the group $\mathbf{g}_1$, while $\{e_{\frac{n}{2}+1}, \cdots, e_n\}$ belong to $\mathbf{g}_2$. Now, consider a CM sketch with a hash function h that maps the events to $w = 200$ bins. Let $N = 10000$ be the number of elements in the data stream. Hence, the additive estimation error is bounded by $\varepsilon_A \leq 50$. Suppose the expected frequencies for group $\mathbf{g}_1$ and $\mathbf{g}_2$ are 1000 and 10, respectively. As a result, the expected-frequency estimation is bounded by 1050 for $\mathbf{g}_1$, resulting in a small error of only 5% for group $\mathbf{g}_1$. On the other hand, this error is as high as 500% for $\mathbf{g}_2$.*

From Example 2, it is clear that the additive error does not provide a strong guarantee independent of the element frequencies. Specifically, providing large estimation errors relative to small frequency values, it is *unfair* for unpopular groups. This motivates us to instead promote **approximation factor** as a stronger frequency estimation guarantee.

Definition 1 (Approximation Factor). *Let $\hat{f}(e)$ be the frequency estimation of the element type e by a count-min sketch CM. The approximation factor of CM for e is* $\alpha(e) = \frac{\hat{f}(e)}{f(e)}$.

Using Definition 1, we define the notion of *group fairness* that equalizes the expected approximation factor across various groups. Specifically, a CM is called fair if it satisfies Definition 2.

Definition 2 (Group-Fair Count-Min (FCM)). *A count-min sketch is group-fair iff:*

$$\forall \mathbf{g}_i, \mathbf{g}_j \in \mathcal{G}, \quad \mathbb{E}_{e \in \mathbf{g}_i}\left[\alpha(e)\right] = \mathbb{E}_{e' \in \mathbf{g}_j}\left[\alpha(e')\right]$$

Without loss of generality and to ease the explanations, we use the binary groups $\mathbf{g}_1$ and $\mathbf{g}_2$ for explaining our sketches and drawing the analyses. Our results readily hold for non-binary and *arbitrary grouping* of element types, as we shall further discuss in Section 3.4.

2.4 Problem Formulation

With the necessary terms and notations defined, we now formally define our problem of interest:

Definition 3. *Given a data stream of elements $\mathcal{D}$ drawn from a universe of element types $\mathcal{U}$ where each element type belongs to a group $g \in \mathcal{G}$, design a group-fair count-min sketch, whose overall additive error increase compared to CM (price of fairness) is small.*

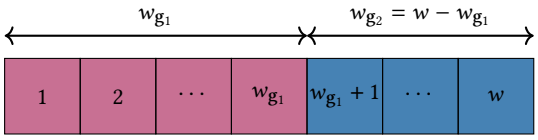

Figure 1: Illustration of a column-partitioning based group-fair min-count with one row, i.e., one hash function $\mathsf{h}(.)$.

2.5 Solution Overview

A main issue that causes the unbounded multiplicative error in CM is that high-frequency elements can collide with low-frequency ones, adding a major overestimation to their counts. Similarly, a sketch in our case might not be group-fair if elements in $\mathbf{g}_1$ collide with elements in $\mathbf{g}_2$.

One way to reduce the likelihood of such collisions is to substantially increase the number of bins w and/or the number of independent hash functions d used in the estimation, i.e., the number of rows in the sketch. However, this approach has a fundamental drawback: to ensure that an element $e \in \mathbf{g}_1$ is unlikely to collide with any element in $\mathbf{g}_2$ across at least one of the d rows, the sketch size must scale linearly with n, the number of distinct element types—defeating the purpose of sketching.

Instead, to design group-fair CM, our goal is to *make it impossible for elements in different groups to collide.* Specifically, we ensure our goal by designing group-aware "semi-uniform" hash schemes that isolate elements of $\mathbf{g}_1$ and $\mathbf{g}_2$ by strategically partitioning and assigning the columns of the sketch to $\mathbf{g}_1$ and $\mathbf{g}_2$ (Section 3). Proving that our sketch satisfies Definition 2, we study its price of fairness (Section 5).

3 Fair-Count-Min using Group-Aware Semi-Uniform Hashing

Our idea for generating group-fair CM (FCM) is to utilize "semi-uniform" hash functions that separate element types of various groups. Specifically, reserving $w_{\mathbf{g}_1}$ bins for the group $\mathbf{g}_1$and $w_{\mathbf{g}_2} = w - w_{\mathbf{g}_1}$ for the group $\mathbf{g}_2$, the semi-uniform hash function $\mathsf{h}(.)$ assigns each element in $\mathbf{g}_1$to the first $w_{\mathbf{g}_1}$ bins and the others to the remaining $w_{\mathbf{g}_2}$ bins (Figure 1):

$$\mathsf{h} : \begin{cases} \mathbf{g}_1 \rightarrow [w_{\mathbf{g}_1}] \\ \mathbf{g}_2 \rightarrow w_{\mathbf{g}_1} + [w_{\mathbf{g}_2}] \end{cases}$$

The key question we shall answer in the rest of this section is whether there exists a value $w_{\mathbf{g}_1}$ (hence $w_{\mathbf{g}_2} = w - w_{\mathbf{g}_1}$) for which a CM based on the semi-uniform hash scheme becomes group-fair.

In the following, first, we focus on the case where $d = 1$, i.e., the sketch is based on only one hash function h. Next, we extend our analysis to the general values of d.

3.1 $d = 1$

First, let us derive the expected approximation factor for an element e in group $\mathbf{g}_1$. For the case of $d = 1$, we employ the stronger guarantee of the average approximation factor instead of the expected one. For clarity, we use the expectation notation $\mathbb{E}$ to also denote the average. Let $n_{\mathbf{g}_1}$ be the number of element types in $\mathbf{g}_1$.

APPENDIX

A Additional Motivation Examples

Example 3: *On platforms like Twitter, hashtags appear as a rapid, high-volume data stream. Automated or troll-bot hashtags can generate millions of occurrences per hour, while rare but potentially critical hashtags may surface only a few thousand times. In a CM, a small additive error can severely distort counts for the rarer tags. Although this may seem like a "free boost", the resulting collision-induced overcounts undermine reliability, making it hard to determine whether a rare hashtag is truly trending or merely inflated by overlap with high-volume tags. As a result, trend detection, resource allocation, and moderation decisions such as filtering become noisier—false signals may emerge while genuinely urgent topics are delayed or misclassified.*

Example 4: *In network monitoring, consider two groups of IPs: malicious and benign IPs. While benign IP addresses (e.g., Google DNS) generate massive traffic, the malicious IPs might only appear sporadically. A CM with a small additive error has a negligible impact on the counts of the benign group. This error, on the other hand, can artificially inflate the counts for the malicious IPs, hiding meaningful anomalies. This makes it harder to detect subtle but critical security threats.*

Example 5: *In online advertising, platforms track how often each ad or user click event occurs. Popular ads may generate millions of impressions, while niche ads may only get a few hundred. With CM, small additive errors hardly affect big advertisers but can completely distort the performance metrics for small advertisers — for example, inflating a niche ad's 50 clicks into 500 due to collisions. This creates unfair reporting, potentially wasting budgets or hiding click fraud.*

B Price of Fairness: Randomness vs. Uniformity

In Section 5, we observed a counterintuitive result for $d = 1$, proving a negative price of fairness (PoF) for FCM, compared to a regular CM sketch that uses a random hash function. This result can be explained by the fact that a random hash function is unlikely to *uniformly* distribute the elements to the buckets (an interesting related topic is *the occupant problem* [59]). As a result, some of the buckets will have more than $\frac{n}{w}$ elements in them. For example, Figure 33 illustrates a random hashing of $n = 100$ element types into $w = 10$ buckets. While a uniform distribution would allocate 10 elements to each bucket, the random hashing allocated up to 18 elements in one bucket.

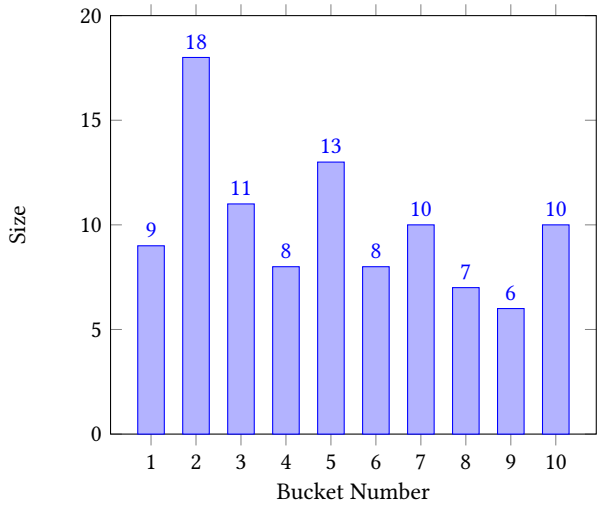

Figure 33: Illustration of a random distribution of $n = 100$ elements to $w = 10$ buckets.

Elements in large buckets will suffer from a large additive error, and there are a large number of them in such buckets. Hence, such buckets significantly increase the total additive error of the CM.

FCM increases the size-uniformity of the buckets by reserving $w_{\mathbf{g}} = \frac{n_{\mathbf{g}}}{n} w$ for each group $\mathbf{g}$. For example, Figure 34 illustrates the allocation of the 100 elements to the 10 buckets, by dividing (arbitrarily) the elements into two groups of size $n_{\mathbf{g}_1} = 50$, $n_{\mathbf{g}_2} = 50$. As a result, each group is allocated 5 buckets, and the distribution of the elements is more uniform, reducing the maximum size of the buckets (in this example) to 14. Increasing the uniformity (reducing the change of large-size buckets) helps to reduce the total additive error of FCM versus CM.

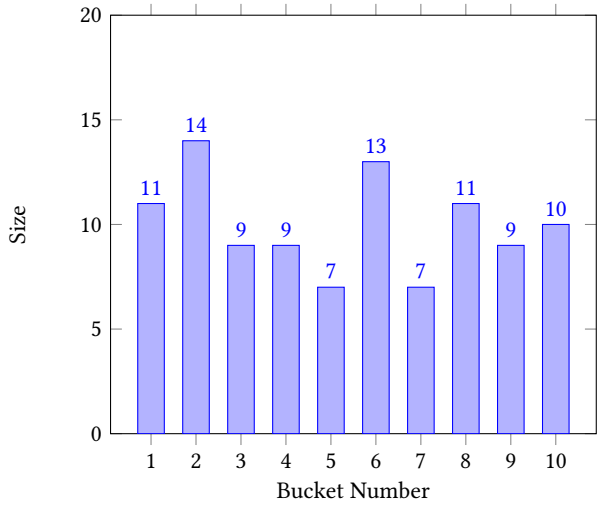

Figure 34: Illustration of a random distribution of $n = 100$ elements with two groups of sizes $n_{\mathbf{g}_1} = 50$ and $n_{\mathbf{g}_2} = 50$ to $w = 10$ buckets, based on FCM.

B.1 PoF for Uniform hashing ($d = 1$)

Using a hashing scheme (such as data-informed hashmaps [52]) that ensures uniformity by allocating exactly $\frac{n}{w}$ elements to each bucket can resolve the non-uniformity issue in the CM sketches. In the following, we compute the PoF of FCM for $d = 1$ when a uniform hashing scheme is used.

Since the hashing is uniform, the size of each bucket i in the CM sketch is

$$C_i = \frac{n}{w}$$

Therefore, each element e_j collides with exactly $\frac{n}{w} - 1$ other elements. In other words, the frequency of each element contributes to the additive error of exactly $\frac{n}{w} - 1$ other elements.

As a result, the total additive error can be computed as

$$\begin{aligned}\mathcal{L}_{CM} &= \sum_{j=1}^{n} \varepsilon_A(e_j) \\ &= \sum_{j=1}^{n} f(e_j)\left(\frac{n}{w} - 1\right) = \left(\frac{n}{w} - 1\right)\sum_{j=1}^{n} f(e_j) \\ &= N\left(\frac{n}{w} - 1\right) \end{aligned} \tag{16}$$

Now, let us compute $\mathcal{L}_{FCM}$, the total additive error for the regular fair-count-min sketch.

$$\mathcal{L}_{FCM} = \sum_{j=1}^{n} \varepsilon_A(e_j) = \sum_{e_j \in \mathrm{g}_1} \varepsilon_A(e_j) + \cdots + \sum_{e_j \in \mathrm{g}_\ell} \varepsilon_A(e_j)$$

Following the same calculation as in Equation 16, for every group $\mathrm{g} \in \mathcal{G}$,

$$\sum_{e_j \in \mathrm{g}} \varepsilon_A(e_j) = N_{\mathrm{g}}\left(\frac{n_{\mathrm{g}}}{w_{\mathrm{g}}} - 1\right)$$

Hence,

$$\mathcal{L}_{FCM} = \sum_{e_j \in \mathrm{g}_1} \varepsilon_A(e_j) + \cdots + \sum_{e_j \in \mathrm{g}_\ell} \varepsilon_A(e_j) = \sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}}\left(\frac{n_{\mathrm{g}}}{w_{\mathrm{g}}} - 1\right)$$

From Theorem 1, we know, $w_{\mathrm{g}} = \frac{n_{\mathrm{g}}}{n} w, \forall \mathrm{g} \in \mathcal{G}$, while $\sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}} = N$. Therefore,

$$\begin{aligned}\mathcal{L}_{FCM} &= \sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}}\left(\frac{n_{\mathrm{g}}}{w_{\mathrm{g}}} - 1\right) \\ &= \sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}}\left(\frac{n_{\mathrm{g}}}{w_{\mathrm{g}}}\right) - \sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}} = \sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}}\left(\frac{n_{\mathrm{g}}}{w_{\mathrm{g}}}\right) - N \\ &= \left(\frac{n}{w}\right)\sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}} - N \qquad \texttt{//since } \frac{n_{\mathrm{g}}}{w_{\mathrm{g}}} = \frac{n}{w}, \forall \mathrm{g} \in \mathcal{G} \\ &= N\left(\frac{n}{w}\right) - N = N\left(\frac{n}{w} - 1\right) \end{aligned} \tag{17}$$

Finally, combining Equations 12, 16, and 17, we get,

$$PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} = N\left(\frac{n}{w} - 1\right) - N\left(\frac{n}{w} - 1\right) = 0 \tag{18}$$

That is, the PoF is zero for $d = 1$, when a uniform hashing scheme is used.

B.2 PoF $d = 1$ vs. $d > 1$ on Synthetic

C Additional Experiment Results

In this section, we present comprehensive experimental results across five datasets: 1) GOOGLE NGRAM, 2) CENSUS, 3) NYC BUS, 4) DDOS, 5)REDDIT-TWITTER. We also provide the absolute values of the approximation factors and additive errors for each group to facilitate a clearer understanding of the trends in unfairness and the associated price of fairness under each setting.

Total Additive Error			n_{g_1} (n = 10,000)								
			9000	**8000**	**7000**	**6000**	**5000**	**4000**	**3000**	**2000**	**1000**
$d = 1$	theoretical	**CM**	19,047,019	28,054,816	37,081,001	46,092,289	55,121,050	64,147,002	73,186,882	82,323,243	91,265,426
		FCM	18,985,224	28,028,987	37,003,399	46,050,674	55,115,538	64,089,586	73,165,549	82,249,909	91,228,281
	empirical	**CM**	19,039,343	28,085,806	37,053,427	46,096,489	55,113,194	64,234,176	73,227,052	82,283,374	91,328,074
		FCM	18,991,509	27,982,573	37,002,246	46,071,394	55,086,890	64,133,815	73,168,189	82,230,649	91,276,271
		PoF	↓-47,834	↓-103,233	↓-51,181	↓-25,095	↓-26,304	↓-100,361	↓-58,863	↓-52,725	↓-51,803
$d = 5$	empirical	**CM**	7,964,348	11,572,548	16,458,026	22,023,181	28,305,699	34,442,308	40,959,703	47,230,290	54,257,770
		FCM	11,695,556	17,114,933	22,666,115	28,053,739	33,856,154	39,448,942	44,913,211	50,089,698	55,893,637
		PoF	↑3,731,208	↑5,542,385	↑6,208,089	↑6,030,558	↑5,550,455	↑5,006,634	↑3,953,508	↑2,859,408	↑1,635,867

Table 2: comparison of CM and FCM w.r.t additive errors and the price of fairness across different n_{g_1} values for $d = 1$ and $d = 5$.

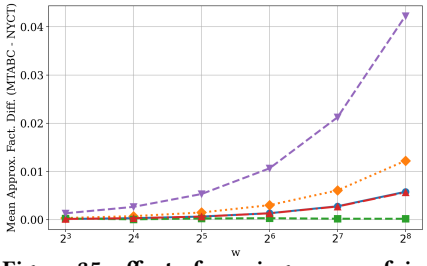

Figure 35: effect of varying w on unfairness, NYC BUS, $d = 5$.

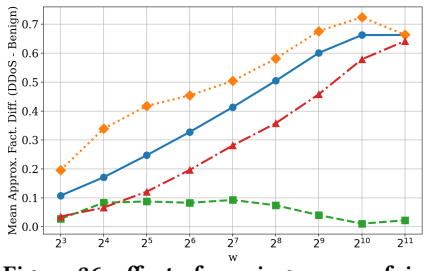

Figure 36: effect of varying w on unfairness, DDOS, $d = 5$.

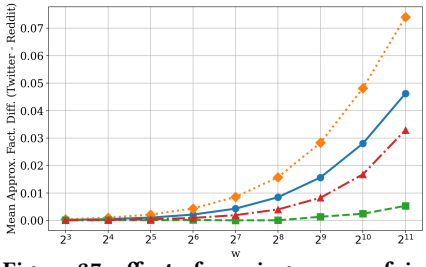

Figure 37: effect of varying w on unfairness, REDDIT-TWITTER, $d = 5$.

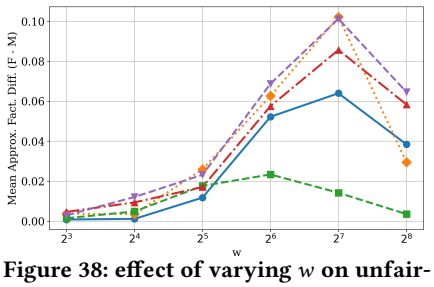

Figure 38: effect of varying w on unfairness, CENSUS, $d = 5$.

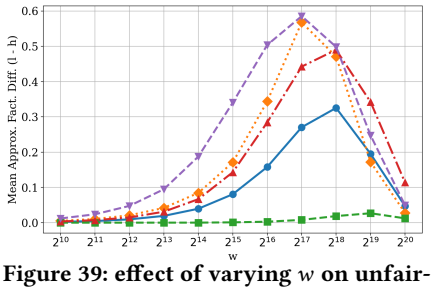

Figure 39: effect of varying w on unfairness, GOOGLE NGRAM, $d = 5$.

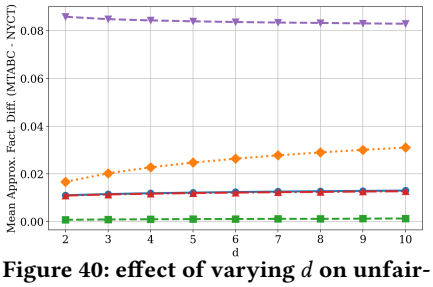

Figure 40: effect of varying d on unfairness, NYC BUS, $w = 512$.

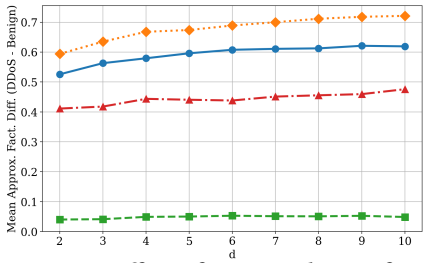

Figure 41: effect of varying d on unfairness, DDOS, $w = 64$.

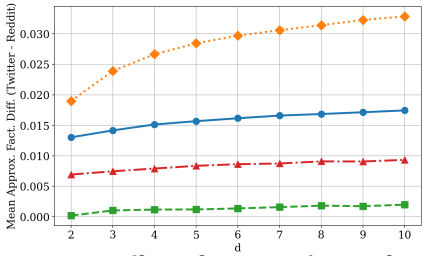

Figure 42: effect of varying d on unfairness, REDDIT-TWITTER, $w = 512$.

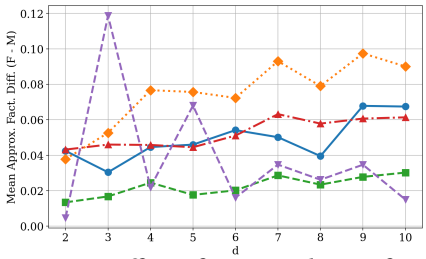

Figure 43: effect of varying d on unfairness, CENSUS, $w = 64$.

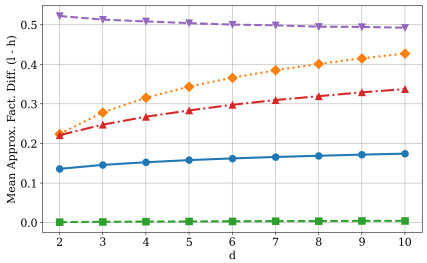

Figure 44: effect of varying d on unfairness, GOOGLE NGRAM, $w = 65536$.

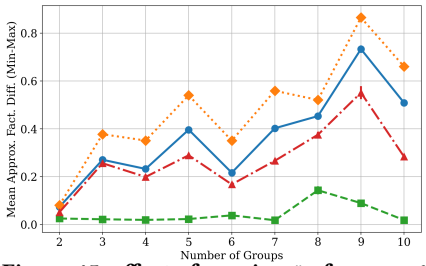

Figure 45: effect of varying # of groups ℓ on unfairness, CENSUS, $w = 64, d = 5$.

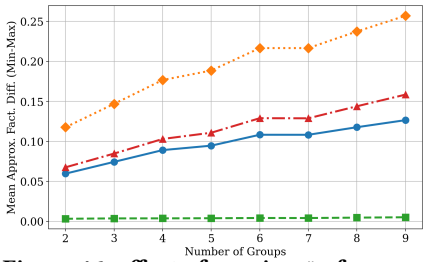

Figure 46: effect of varying # of groups ℓ on unfairness, GOOGLE NGRAM, $w = 65536, d = 5$.

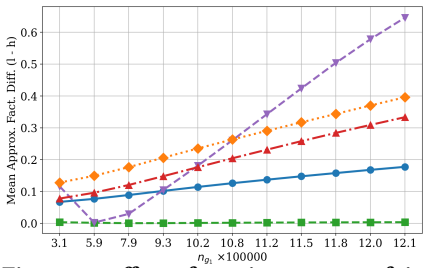

Figure 47: effect of varying n_{g_1} on unfairness, GOOGLE NGRAM, $w = 65536, d = 5$.

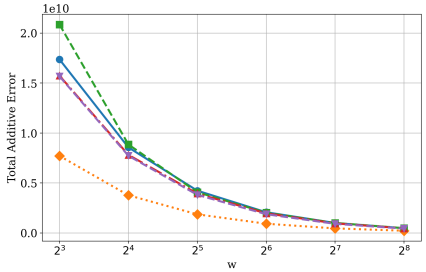

Figure 48: effect of varying w on price of fairness, NYC BUS, $d = 5$.

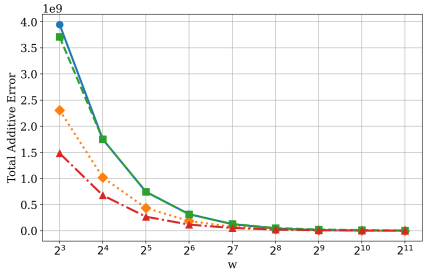

Figure 49: effect of varying w on price of fairness, DDOS, $d = 5$.

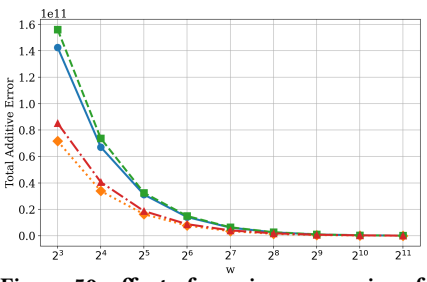

Figure 50: effect of varying w on price of fairness, REDDIT-TWITTER, $d = 5$.

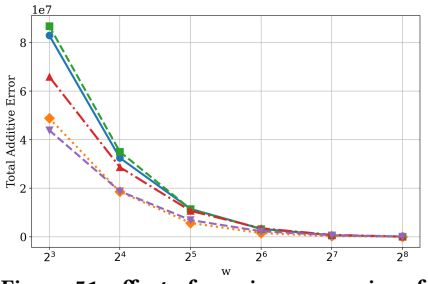

Figure 51: effect of varying w on price of fairness, CENSUS, $d = 5$.

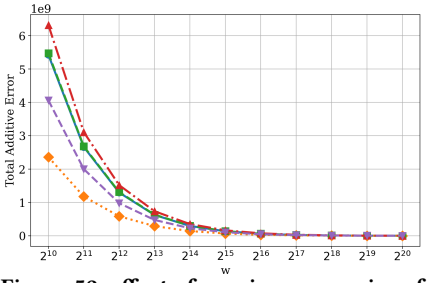

Figure 52: effect of varying w on price of fairness, GOOGLE NGRAM, $d = 5$.

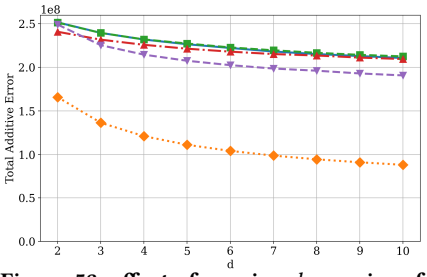

Figure 53: effect of varying d on price of fairness, NYC BUS, $w = 512$.

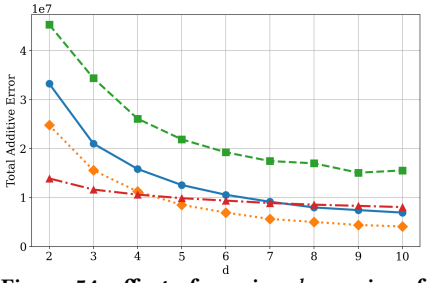

Figure 54: effect of varying d on price of fairness, DDOS, $w = 64$.

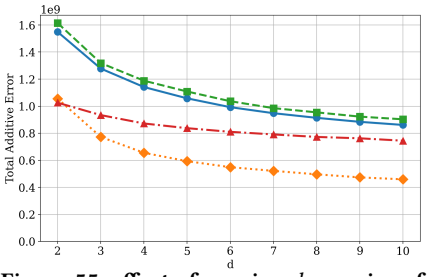

Figure 55: effect of varying d on price of fairness, REDDIT-TWITTER, $w = 512$.

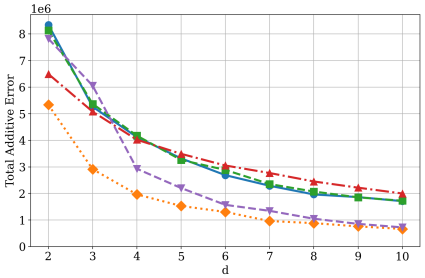

Figure 56: effect of varying d on price of fairness, CENSUS, $w = 64$.

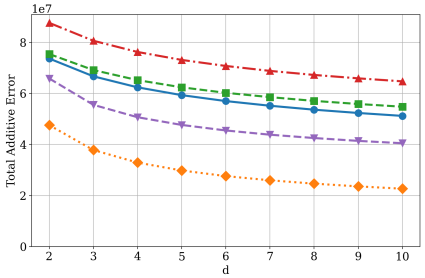

Figure 57: effect of varying d on price of fairness, GOOGLE NGRAM, $w = 65536$.

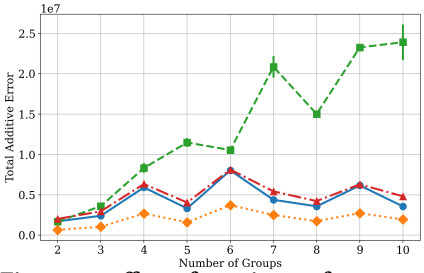

Figure 58: effect of varying # of groups ℓ on price of fairness, CENSUS, $w = 64$, $d = 5$.

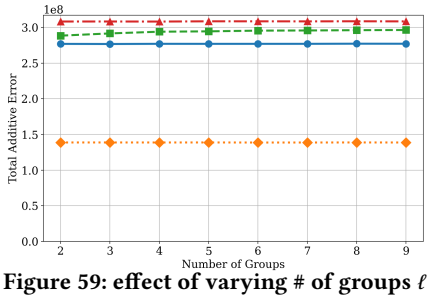

Figure 59: effect of varying # of groups ℓ on price of fairness, GOOGLE NGRAM, $w = 65536$, $d = 5$.

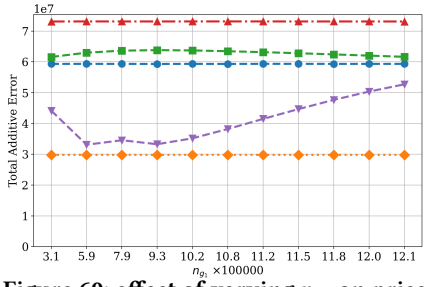

Figure 60: effect of varying n_{g_1} on price of fairness, GOOGLE NGRAM, $w = 65536$, $d = 5$.

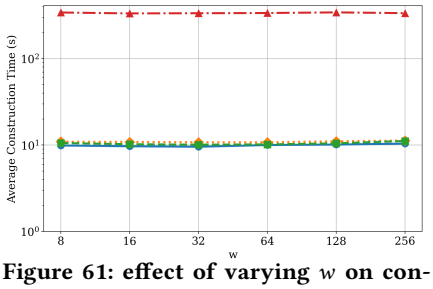

Figure 61: effect of varying w on construction time, NYC BUS, $d = 5$.

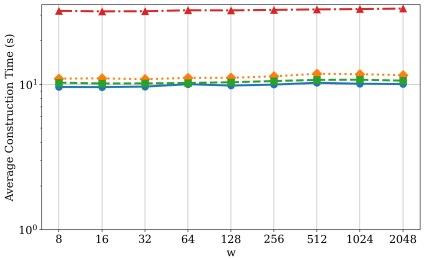

Figure 62: effect of varying w on construction time, DDOS, $d = 5$.

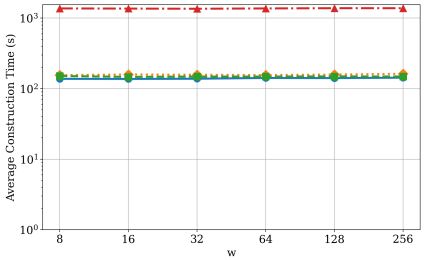

Figure 63: effect of varying w on construction time, REDDIT-TWITTER, $d = 5$.

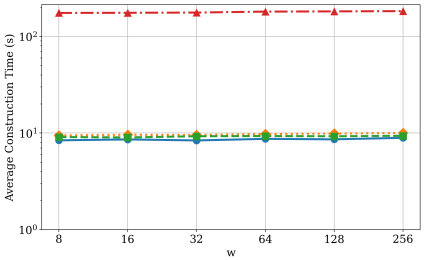

Figure 64: effect of varying w on construction time, CENSUS, $d = 5$.

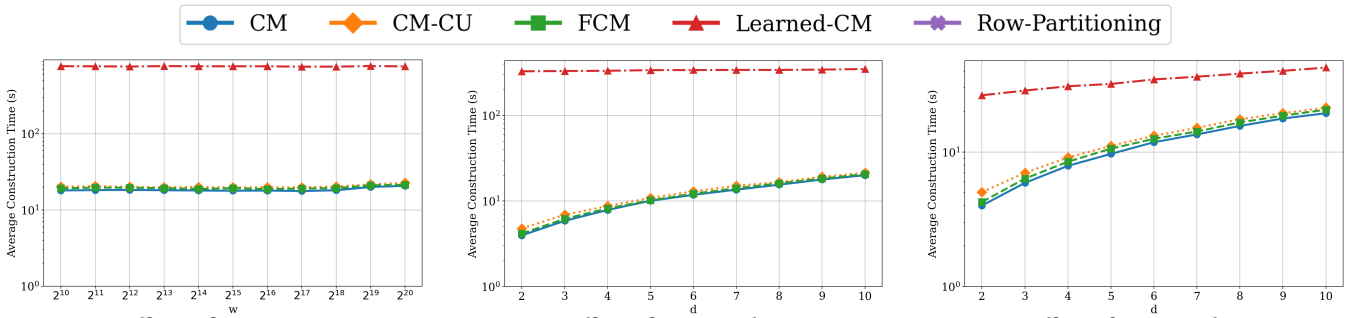

Figure 65: effect of varying w on construction time, GOOGLE NGRAM, $d = 5$.

Figure 66: effect of varying d on construction time, NYC BUS, $w = 512$.

Figure 67: effect of varying d on construction time, DDOS, $w = 64$.

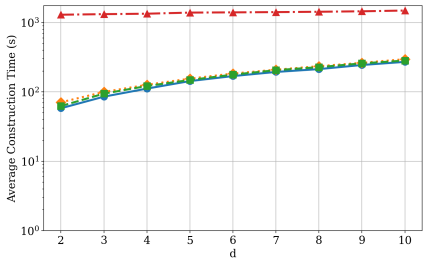

Figure 68: effect of varying d on construction time, REDDIT-TWITTER, $w = 512$.

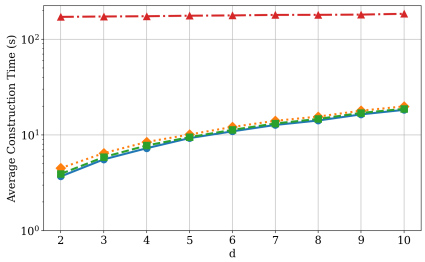

Figure 69: effect of varying d on construction time, CENSUS, $w = 64$.

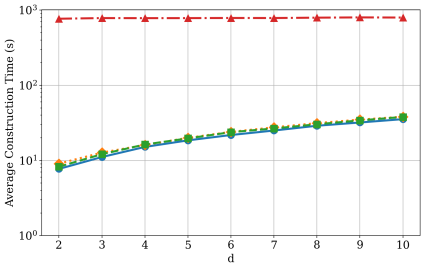

Figure 70: effect of varying d on construction time, GOOGLE NGRAM, $w = 65536$.

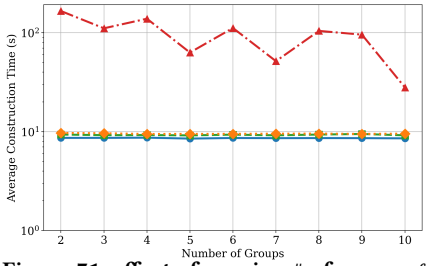

Figure 71: effect of varying # of groups ℓ on construction time, CENSUS, $w = 64$, $d = 5$.

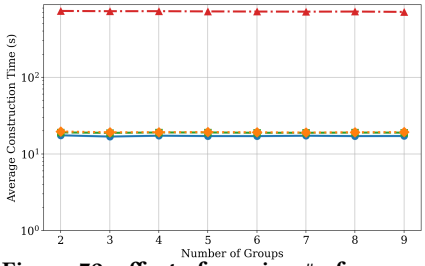

Figure 72: effect of varying # of groups ℓ on construction time, GOOGLE NGRAM, $w = 65536$, $d = 5$.

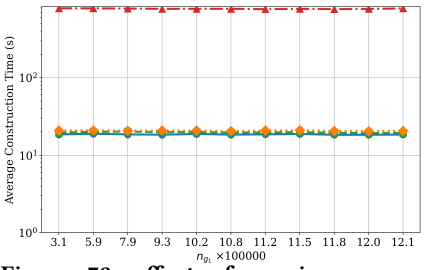

Figure 73: effect of varying n_{g_i} on construction time, GOOGLE NGRAM, $w = 65536$, $d = 5$.

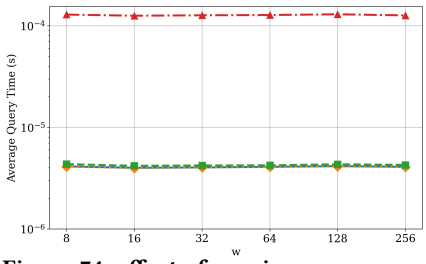

Figure 74: effect of varying w on query time, NYC BUS, $d = 5$.

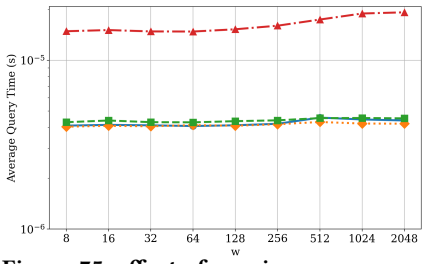

Figure 75: effect of varying w on query time, DDOS, $d = 5$.

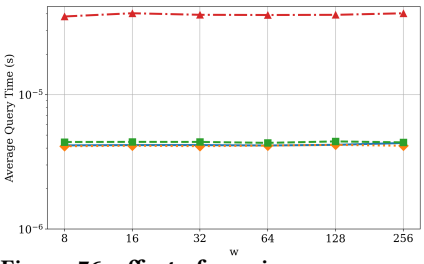

Figure 76: effect of varying w on query time, REDDIT-TWITTER, $d = 5$.

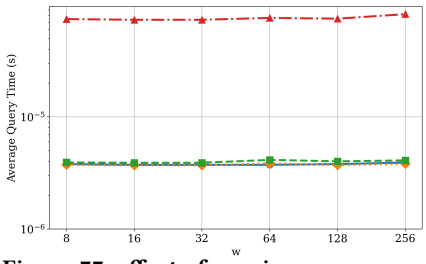

Figure 77: effect of varying w on query time, CENSUS, $d = 5$.

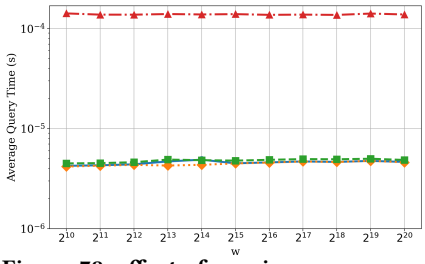

Figure 78: effect of varying w on query time, GOOGLE NGRAM, $d = 5$.

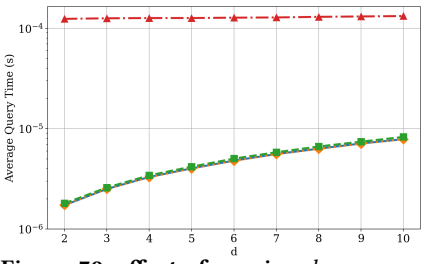

Figure 79: effect of varying d on query time, NYC BUS, $w = 512$.

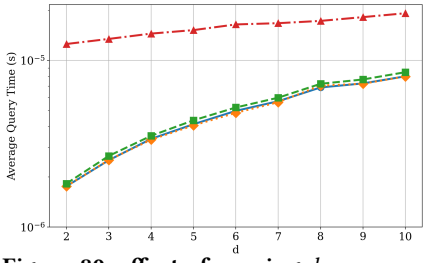

Figure 80: effect of varying d on query time, DDOS, $w = 64$.

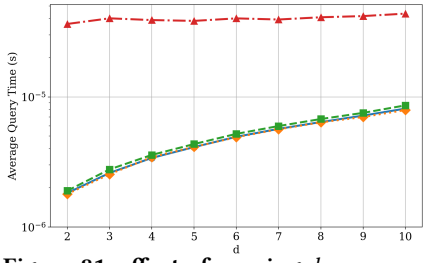

Figure 81: effect of varying d on query time, REDDIT-TWITTER, $w = 512$.

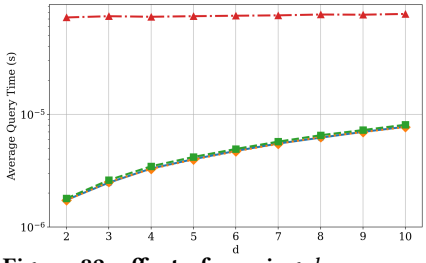

Figure 82: effect of varying d on query time, CENSUS, $w = 64$.

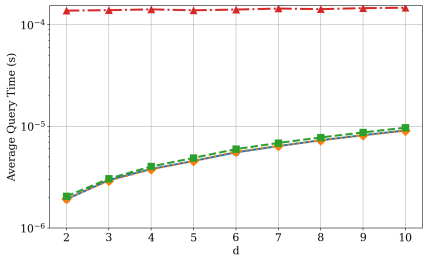

Figure 83: effect of varying d on query time, GOOGLE NGRAM, $w = 65536$.

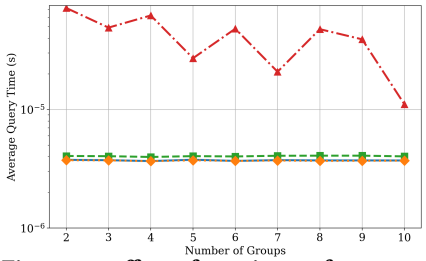

Figure 84: effect of varying # of groups ℓ on query time, CENSUS, $w = 64, d = 5$.

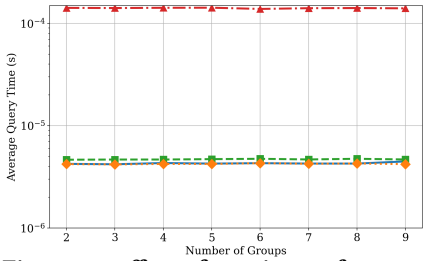

Figure 85: effect of varying # of groups ℓ on query time, GOOGLE NGRAM, $w = 65536, d = 5$.

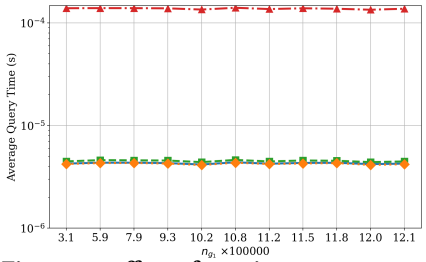

Figure 86: effect of varying n_{g_1} on query time, GOOGLE NGRAM, $w = 65536, d = 5$.