Summarizing and Mining Skewed Data Streams

Graham Cormode[*] S. Muthukrishnan[†]

Abstract

Many applications generate massive data streams. Summarizing such massive data requires fast, small space algorithms to support post-hoc queries and mining. An important observation is that such streams are rarely uniform, and real data sources typically exhibit significant skewness. These are well modeled by Zipf distributions, which are characterized by a parameter, z, that captures the amount of skew.

We present a data stream summary that can answer point queries with ε accuracy and show that the space needed is only $O(\varepsilon^{-\min\{1,1/z\}})$. This is the first $o(1/\varepsilon)$ space algorithm for this problem, and we show it is essentially tight for skewed distributions. We show that the same data structure can also estimate the L_2 norm of the stream in $o(1/\varepsilon^2)$ space for $z > \frac{1}{2}$, another improvement over the existing $\Omega(1/\varepsilon^2)$ methods.

We support our theoretical results with an experimental study over a large variety of real and synthetic data. We show that significant skew is present in both textual and telecommunication data. Our methods give strong accuracy, significantly better than other methods, and behave exactly in line with their analytic bounds.

Keywords: data stream analysis, data mining, Zipf distribution, power laws, heavy hitters, massive data.

1 Introduction

A number of applications—real-time IP traffic analysis, managing web clicks and crawls, sensor readings, email/SMS/blog and other text sources—are instances of massive data *streams*. Here new data arrives very rapidly and often we do not have the space to store all the data. Hence, managing such streams needs algorithmic methods that support *fast updates* and have a *small footprint* of space. See [17] for a detailed motivation for these constraints in the context of IP traffic analysis. Similar motivation can be found in high performance web data analysis [25], mining email streams [40], aggregating sensor data [39], analyzing financial time series [52], processing high-volume scientific measurements [30], detecting communities in communication graphs [7] and others.

There are typically two aspects to analyzing data streams. The first is *summarizing* data streams for post-hoc queries. Data stream methods use a *synopsis* to summarize the data stream on the fly. "Synopsis" means a small space representation of the data stream that can be rapidly updated as the data stream unravels; typical synopses are samples and sketches. The second aspect of analyzing data streams is the type of queries that are supported, whether using synopses or without. Since many of these data stream applications tend to be about monitoring data sources say for adverse events such as intrusion detection, fraudulent communities, etc., an essential set of problems are of the data *mining* genre. For example, how to find heavy hitters or frequent items, large changes and evolving trends, or perform clustering and similarity searching, decision tree classification, regression and other statistical analyses. See the ensemble of websites for project descriptions [41, 26, 48, 1, 15, 28, 45], bibliographies [50], tutorials [20, 34], surveys [5, 24, 44] in addition to a number of special issues in journals, workshops, etc.

We study both summarization and mining problems with data streams. Our work here was initiated by our experience with IP traffic analysis using Gigascope [15], AT&T's IP traffic analysis system, as well as our work on mining text streams [40]. Our observation was that data streams involved distributions that were not arbitrary, but rather typically quite *skewed*. For example, if one studied the distribution of the IP addresses that used a link of the backbone network or the distribution of flows or bytes sent by each IP address, the distribution was zipfian, fractal or multi-fractal. This has been studied in detail in [33]. Similarly, the word frequencies in natural text is well known to be zipfian; word frequencies in text streams such as email or blogs tend to be heavy-tailed [36]. In nature, zipfian distributions (also known as power laws) abound in citation distributions to web accesses for different sites, file sizes transferred over the Internet, etc [8].

Motivated by our experience and the widely-prevalent evidence of skew in data streams, we study summarization and mining problems with skewed data streams. We take a formal approach, focusing on zipfian (more generally, *Zipf-like* which we define later and which is quite general) skew on the domain $1 \ldots U$. The Zipf distribution is also known

[*] `cormode@bell-labs.com` Work completed while author was at Center for Discrete Mathematics and Computer Science (DIMACS) Rutgers University, Piscataway NJ. Supported by NSF ITR 0220280 and NSF EIA 02-05116.

[†] `muthu@cs.rutgers.edu` Division of Computer and Information Systems, Rutgers University. Supported by NSF CCR 0087022, NSF ITR 0220280, NSF 0430852 and NSF EIA 02-05116.

as the Pareto distribution, and is essentially identical to so-called "power-laws" [2] with transformation of parameters. We work in well-established data stream algorithm model and study *probabilistic, approximate* algorithms, that is algorithms that provide ε approximation with probability of success at least $1 - \delta$. As is usual, we study the trade-off between space used by the synopsis and the time per new data update versus the quality of estimations given by ε and δ. Our *theoretical contributions* are as follows.

1. We present a synopsis that uses space $O(\varepsilon^{-\min\{1,1/z\}} \ln 1/\delta)$ for (ε, δ) point queries on z-skewed data streams.

   For $z > 1$, the space used is $o(1/\varepsilon)$; this is the first known $o(1/\varepsilon)$ space algorithm known for *any* synopsis—sample or sketch based—known for such problems. This is of interest since the $O(1/\varepsilon)$ space bound has long been taken as the gold-standard target for data stream algorithm design.

   We can use this synopsis for a variety of mining tasks such as finding heavy-hitters, frequent items for association rule mining, finding significant changes from one time to another, significant differences between different streams, estimating wavelet decomposition of data stream, and so on. In all these cases, our methods improve the $O(1/\varepsilon)$ factor in previously known algorithms to $O(\varepsilon^{-\min\{1,1/z\}})$ for space usage.

2. We prove a matching lower bound, that is, we show that any $(\varepsilon, O(1))$ algorithm for point queries needs at least $\Omega(\varepsilon^{-\min\{1,1/z\}})$ space.

3. We extend our synopsis to estimate the second frequency moment, ie., the sum of squares of the frequencies of items in the data stream. It is equivalently (the square of) the L_2 norm of the vector of frequencies of items in the data stream. The space used is $O(\varepsilon^{\frac{-4}{1+2z}})$ for $1/2 < \varepsilon \leq 1$ and $O(\varepsilon^{\frac{-2}{1+z}})$ for $z > 1$.

   For $z \geq 1/2$, the space used is $o(1/\varepsilon^2)$; all previously known algorithms in contrast use at least $\Omega(1/\varepsilon^2)$ space. Our bound above is additionally interesting because the synopsis methods for L_2 norm are implementations of the Johnson-Lindenstrauss lemma [29]. The lemma states that a set of vectors in Euclidean space can each be replaced by a vector in $O(\frac{1}{\varepsilon^2})$-dimensional space, and inter-vector distances are preserved up to a $(1 \pm \varepsilon)$ factor with high probability. This dependency on ε is tight since a lower bound of $\Omega(\frac{1}{\varepsilon^2})$ has recently been shown [51] for general distributions. Our results show that for skewed data, this lower bound can be avoided.

   We can use this result for a variety of mining tasks. For example, anomaly detection methods in IP traffic analysis use the second frequency moment [35]. Also, it is used as a subroutine in counting subgraphs in massive web graphs in [6], and for quantiles, histograms and other statistical descriptors of the data stream [21].

Our synopsis above is a sketch, ie., inner products of the input distribution with certain random vectors. There are many known sketches [4, 10, 23, 49, 13]. Here, we adopt the Count-Min sketch [13] which dominates all other sketches in terms of space and update time needed to guarantee $\varepsilon\|\boldsymbol{a}\|$ accuracy. Our main theoretical contribution here is to *analyze* estimation methods based on this sketch and prove improved bounds on space usage without compromising any of the other parameters, ie., update time and accuracy. Our algorithms are essentially *oblivious* of the skew value, z but our analysis is skew-aware. We can approach this in two ways: given a desired error bound ε and bound on the skew z, we can allocate space to the sketch as a function of these parameters; alternatively, we can allocate a fixed amount of space for the sketch, and based on the observed skew z, give tight bounds on the worst case error as a function of z. In the latter case irrespective of the data distribution, a simple analysis gives a universal bounds on the error, and using the skew of the distribution we can give tighter bounds.

Our results give additional evidence that CM sketch is versatile and suited for a variety of problems under a range of data distributions. This, coupled with their proven performance within the operational AT&T's IP traffic analysis tool Gigascope [11] at the rate of OC48 links, makes our methods for skewed data stream summarization and mining suitably efficient for real-life data stream management systems in practice.

Our *experimental contributions* are as follows. We consider large streams of both real and synthetic data. We observe that all the real data we consider, from IP network and phone call data to "blogs" and Shakespeare's plays, exhibit significant skew to varying degrees, and our methods capitalize on this. Not only do they outperform other methods, but they behave closely in accordance with our stated bounds. The correlation is sufficiently good that not only can we compare our method to that predicted by our theory, but also we can use our results to compute the skewness of the data with high accuracy. Our conclusion is that by understanding and building skew into our model of data streams, we can realize much stronger results, both in terms of theoretical analysis and practical performance.

1.1 Map.

We give preliminaries in Section 2, then define the CM Sketch in Section 3. We discuss skewed distributions and related work in Section 4. Section 5 gives our results for Point Queries, and Section 6 those for L_2 norm estimation. Our experimental study on a mixture of real and synthetic data is reported in Section 7.

2 Model and Queries

We consider a vector $\boldsymbol{a}$, which is presented in an implicit, incremental fashion. This vector has dimension n, and its current state at time t is $\boldsymbol{a}(t) = [a_1(t), \ldots a_i(t), \ldots a_n(t)]$. For convenience, we shall usually drop t and refer only to the current state of the vector. Initially, $\boldsymbol{a}$ is the zero vector, $\mathbf{0}$, so $a_i(0)$ is 0 for all i. Updates to individual entries of the vector are presented as a stream of pairs. The tth update is (i_t, c_t), meaning that

$$\begin{aligned} a_{i_t}(t) &= a_{i_t}(t-1) + c_t \\ a_{i'}(t) &= a_{i'}(t-1) \qquad i' \neq i_t \end{aligned}$$

We assume throughout that although values of a_i increase and decrease with updates, each $a_i \geq 0$. Our results all generalize to the case where a_is can be less than 0, with small factor increases in space, but we omit details of these extensions for simplicity of exposition.

It is easy to see how this model maps to the motivating data stream applications. For example, for the IP traffic case, each new IP packet with source IP address s and size of the packet p may be seen as updating $a[s] \leftarrow a[s] + p$ to count the total size of flows from source IP address s. Similarly, in the text streaming application, when new text input such as email arrives, we can parse it into words and track word usage frequency in order to track frequent and recently popular ("bursty") words, to attribute authorship based on usage patterns, etc. Here, each new text input updates many new $a[w]$'s for different words or phrases w in the input. See [44] for more examples. We consider two particular types of queries for summarization and mining.

- **Point query.** A point query is, given i, to return an estimate of a_i. Our goal is to give (ε, δ) approximations: the answer should be correct to within additive error of $\varepsilon\|\boldsymbol{a}\|_1$ with probability at least $1 - \delta$. We will analyze the space required as a function of ε and δ required to achieve this.

- **Second Frequency Moment and L_2 Norm** The L_2 norm of a vector, $\|\boldsymbol{a}\|_2$, is defined as $(\sum_i a_i^2)^{\frac{1}{2}}$. The goal is to estimate this within additive error of $\varepsilon\|\boldsymbol{a}\|_2$ (equivalently, with relative error $1 + \varepsilon$) with probability at least $1 - \delta$. The second frequency moment in our model is the square of the L_2 norm, $\|\boldsymbol{a}\|_2^2$.

These two queries appear to be abstract, but they have many concrete applications in a number of mining problems on data streams. Point queries can be used for estimating frequent items for association rule mining [42], heavy hitters[1] [12], significant differences [10] and significant relative changes [14], etc. The L_2 norm estimation is useful in anomaly detection [35], counting triangles in massive web graphs in [6], and for quantiles [23], wavelets [22], histograms [21] and other statistical descriptors of the data stream. They are also useful for partitioning data stream into multiple zones of interest [16]. There is a maturing theory of data stream algorithms with many such applications. Rather than list these applications and show the improvements obtained by using our methods we focus on these primary queries and demonstrate the nature of our improvements in depth.

[1]The heavy hitters problem is to find all i such that $a_i \geq \|\boldsymbol{a}\|_1/k$ for some constant k.

3 The CM Sketch

Many sketches are known [4, 10, 49, 13]. Here, we briefly recap the data structure that is used throughout. The important property is that, given the parameters of the sketch structure, the update procedure is the same no matter what the ultimate query operations are.

The CM sketch is simply an array of counters of width w and depth d, $count[1,1] \ldots count[d,w]$. Each entry of the array is initially zero. Additionally, d hash functions

$$h_1 \ldots h_d : \{1 \ldots n\} \rightarrow \{1 \ldots w\}$$

are chosen uniformly at random from a pairwise-independent family. Once w and d are chosen, the space required is fixed as the wd counters and the d hash functions (which can each be represented in $O(1)$ machine words [43]).

Update and Query Procedure. When an update (i_t, c_t) arrives, meaning that item a_{i_t} is updated by a quantity of c_t, then c_t is added to one count in each row; the counter is determined by h_j. Formally, we set

$$\forall 1 \leq j \leq d : count[j, h_j(i_t)] \leftarrow count[j, h_j(i_t)] + c_t$$

The query procedure is similar: given a query point i, return $\min_{1 \leq j \leq d} count[h, h_j(i)]$ as the estimate. In [13], it was shown that the error for point queries, irrespective of the distribution, is $\varepsilon\|\boldsymbol{a}\|_1 = e/w\|\boldsymbol{a}\|_1$ with probability $1 - \delta = 1 - e^{-d}$. Hence, in order to get ε approximation with probability $1 - \delta$ for point queries, we need $w = e/\varepsilon$ and $d = \log(1/\delta)$.

4 Skew in Data Stream Distributions

In almost every practical setting, the distribution of frequencies of different items displays some amount of skew. Throughout, we will use the popular Zipf distribution to model skewed distributions. The Zipf distribution accurately captures a large number of natural distributions. It was introduced in the context of linguistics, where it was observed that the frequency of the ith most commonly used word in a language was approximately proportional to $1/i$ [53]. Zipf distributions are equivalent to Pareto distributions and power-laws [2].

Formally, a Zipf distribution with parameter z has the property that f_i, the (relative) frequency of the ith most frequent item is given by $f_i = \frac{c_z}{i^z}$, where c_z is an appropriate scaling constant. We will consider distributions over the range $[1 \dots U]$, where U is the range, or universe size. For the skewed distributions we consider, we can often allow U to be ∞. c_z is determined by z (and U) since for a probability distribution we must have $\sum_{i=1}^{U} f_i = 1$. Given a vector $\boldsymbol{a}$ whose entries are distributed according to a Zipf distribution, the count of the ith most frequent item is simply $\|\boldsymbol{a}\|_1 f_i$.

Many skewed distributions are well captured by Zipf distributions with appropriate parameters. Natural phenomena, such as sizes of cities, distribution of income, social networks and word frequency can all be modeled with Zipf distributions. Even the number of citations of papers demonstrates a highly skewed Zipf distribution [47]. More relevant to our study of large data streams, web page accesses for different sites have been observed to obey a skewed Zipf distribution with parameter between 0.65 and 0.8. [9]. The "depth" to which surfers investigate websites is also captured by a Zipf distribution, with parameter 1.5 [27]. Files communicated over the Internet display Zipf distribution in a variety of ways: transmission times are Zipf with parameter approximately 1.2; the size of files requested, transmitted, and available for download are all Zipf with parameters respectively measured as 1.16, 1.06 and 1.06 [8]. FTP traffic sizes was estimated to have z in the range 0.9 to 1.1. More strongly, such skewed behavior of requests appears not only over individual addresses but also when grouped into subnets or larger networks [33], meaning that the skewed distribution is self-similar (multi-fractal).

Related work on Mining Skewed Streams. A distinguishing element of our work is to bring the skew of the data into the analysis of summarizing and mining data whereas much of the extant work deals with arbitrary distributions (with some exceptions). For the heavy hitters problem Manku and Motwani [42] presented the "lossy counting" algorithm that requires space $O(\frac{1}{\varepsilon} \log \varepsilon \|\boldsymbol{a}\|_1)$ to give the same accuracy bounds as our results in general; but under the assumption that each new item is drawn from a fixed probability distribution, then the space is (expected) $O(\frac{1}{\varepsilon})$ and the error guaranteed. Our results are dual to this, given guaranteed space bounds and expected error bounds; however, with more information about the distribution, our bounds are dependent on skew z, being much better for moderate to large skew, but never worse. For the top k problem, [10] specifically studied Zipfian data and showed that for $z > \frac{1}{2}$, $O(\frac{k}{\varepsilon^2})$ space suffices. For large skew, our methods improve this bound to $O(\frac{k}{\varepsilon})$. Using data skew is not uncommon in database research, but only recently there are examples of data mining in presence of skew in massive data such as [18] of analyzing trading anomalies. Our work differs from previous works by being a systematic algorithmic study of summarization and mining problems in data streams with skew to give much improved bounds and performance.

Zipf tail bounds. For our analysis, we will divide up the range of the parameter z into three regions. We refer to $\frac{1}{2} < z \leq 1$ as *moderate skew*, and $1 < z$ as *skewed*. Otherwise, when $z \leq \frac{1}{2}$, we will say that the distribution has *light skew*.

The following facts result from bounding the discrete distribution by its continuous counterpart.

FACT 4.1. *For $z > 1$, $1 - \frac{1}{z} \leq c_z \leq z - 1$.*

FACT 4.2. *For $z > 1$,*

$$\frac{c_z k^{1-z}}{z-1} \leq \sum_{i=k}^{U} f_i \leq \frac{c_z (k-1)^{1-z}}{z-1}$$

FACT 4.3. *For $z > \frac{1}{2}$,*

$$\frac{c_z^2 k^{1-2z}}{2z-1} \leq \sum_{i=k}^{U} f_i^2 \leq \frac{c_z^2 (k-1)^{1-2z}}{2z-1}$$

Our analyses generalize to when the data distribution is dominated by zipfian or more generally, what we call *Zipf-like* distributions: a distribution is *Zipf-like* with parameter $z > 1$ if the tail after k largest items has weight at most k^{1-z} of the total weight (one could also allow scaling by a constant, eg, the tail has weight at most ck^{1-z}; such extensions follow easily). Although we state results in this paper for zipfian data, with a few more technical details, the results hold for Zipf-like distributions as well.

5 Point Queries

5.1 Upper Bounds

The crucial insight for giving better bounds for the Count-Min sketch in the presence of skewed distributions is the observation that items with large counts can cause our estimates to be poor if they collide with other items in the array of counters. In a skewed distribution a few items consume a large fraction of the total count. When estimating the count of an item, if none of these large items collide with it under the hash functions, then we can argue that the estimates will be better than the $w = 1/\varepsilon$ bound given by the generic argument in [13].

THEOREM 5.1. *For a Zipf distribution with parameter z, the space required to answer point queries with error $\varepsilon\|\boldsymbol{a}\|_1$ with probability at least $1 - \delta$ is given by $O(\varepsilon^{-\min\{1,1/z\}} \ln 1/\delta)$.*

Proof. For $z \leq 1$, the best results follow from analysis in [13]. For $z > 1$, we use the same estimation technique to return an estimate for a_i as $\hat{a}_i = \min_j count[h_j(i)]$, but give a new analysis. The estimate returns a_i plus some additional "error" coming from the counts of items that collide with i under the hash functions. We split the error in our estimate into two parts: (i) collisions with some of the largest items and (ii) noise from the non-heavy items. If the sketch has width w, then let $k = w/3$. With constant probability ($\frac{2}{3}$) over the choice of hash functions, none of the k heaviest items collide with the point we are testing in any given row.

The expectation of the estimate for i is

$$a_i + \frac{1}{w} \sum_{x=k+1, x \neq i}^{U} a_x \leq a_i + \|\boldsymbol{a}\|_1 k^{1-z}/w. \tag{5.1}$$

This uses Fact 4.2 from Section 4 to bound the weight of the tail of the Zipf distribution after the k largest items are removed. Setting $k^{1-z}/w = \varepsilon/3$ and recalling that $w = 3k$ leads us to choose $w = 3k = 3(\frac{1}{\varepsilon})^{1/z}$. We can now apply the Markov inequality to show that the error is bounded by $\varepsilon\|\boldsymbol{a}\|_1$ with probability at least $1 - \frac{1}{3} - \frac{1}{3} = \frac{1}{3}$. This applies to each estimate; since we take the minimum of all the estimates, then this probability is amplified to $1 - \frac{2}{3}^d$ over the d separate estimations. □

5.2 Lower Bounds

We now present lower bounds for the space required to answer point queries, which shows that our analysis above is asymptotically tight (since [13] shows the CM sketch data structure gives ε error over general distributions with $O(\frac{1}{\varepsilon})$ space).

THEOREM 5.2. *The space required to answer point queries correctly with any constant probability and error at most* $\varepsilon\|\boldsymbol{a}\|_1$ *is* $\Omega(\varepsilon^{-1})$ *over general distributions, and* $\Omega(\varepsilon^{-1/z})$ *for Zipf distributions with parameter z, assuming* $n = \Omega(\varepsilon^{-\min\{1,1/z\}})$.

Proof. Our proof relies on a reduction to the Index problem in communication complexity. There are two players, A and B. A holds a bitstring of length n, and is allowed to send a message to B who wishes to compute the ith bit of the bitstring. Since B is not allowed to communicate with A, then any protocol to solve this problem, even probabilistically, requires $\Omega(n)$ bits of communication [37]. We will reduce to this problem by encoding a bitstring in such a way that if we could answer point queries with sufficient accuracy, we could recover bits from the bitstring. This is sufficient to show a lower bound on the size of the data structure required to answer such queries.

For general distributions, we take a bitstring $B[1 \ldots \frac{1}{2\varepsilon}]$ of length $n = \frac{1}{2\varepsilon}$ bits, and create a set of counts. We set $a_i = 2$ if $B[i] = 1$. Otherwise, we set $a_i = 0$ otherwise, and add 2 to a_0. Now, observe that whatever the value of B, $\|\boldsymbol{a}\|_1 = 1/\varepsilon$. If we can answer point queries with accuracy $\varepsilon\|\boldsymbol{a}\|_1 = 1$, then we can test any a_i and determine the value of $B[i]$ by reporting 1 if the estimated value of a_i is above εN, and 0 otherwise. Therefore, the space used must be at least $\Omega(\frac{1}{\varepsilon})$ bits.

The same idea applies when we restrict ourselves to Zipf distributions. However, the counts must follow the Zipf pattern. We again encode a bitstring B, this time using the k largest counts from the Zipf distribution. Now we set $a_{2i} = f_i N$ (for some suitably large value of N) if $B[i] = 1$, else we set $a_{2i+1} = f_i N$ if $B[i] = 0$. This time, we can recover the first k bits of B provided that $f_k \geq 2\varepsilon$: if f_k is less than this, then the error in approximation does not allow us to distinguish this value from zero. Using the bounds on f_i for skewed Zipf distributions, we have $f_k = \frac{c_z}{k^z} \geq 2\varepsilon$. To get the best lower bound, we choose k as large as possible subject to these constraints, Solving for k, we find $k = \frac{c_z}{2}^{1/z} \frac{1}{\varepsilon}^{1/z}$. The term $\frac{c_z}{2}^{1/z}$ is bounded below by $(z-1)/2$ for $1 < z \leq 2$, and may be treated as a constant. Thus, k is fixed as $c\frac{1}{\varepsilon}^{1/z}$ bits of B for some constant c. This results in the stated space bounds by again appealing to the Index problem. □

5.3 An example application: Top-k items

As mentioned earlier, supporting point queries post-hoc on data stream synopsis has many applications. Here, we focus on describing one of them.

A common query for a variety of management and analysis settings is to find the top-k: for example, find the top 100 users of bandwidth on a network, or find the top 10 new terms in a message stream. Such queries can be answered by point queries, by tracking the most frequent items that are seen as the stream unravels. We need to choose the parameter ε appropriately: too large, and we will not be able to answer the query with sufficient accuracy, and the results may be unreliable. When the distribution is skewed, we can apply our above results and give very tight bounds.

To give the correct answer, we need to bound the error by εa_k (where, here, we use a_k to denote the frequency of the kth most frequent item in $\boldsymbol{a}$) instead of $\varepsilon\|\boldsymbol{a}\|_1$. Using the above analysis for the expectation of the error in the estimation of any frequency from equation (5.1), we set the expected error equal to εa_k:

$$\frac{\|\boldsymbol{a}\|_1 k^{1-z}}{w} = \frac{\varepsilon\|\boldsymbol{a}\|_1 k^{-z}}{2}$$

and so $w = O(\frac{k}{\varepsilon})$ for $z > 1$. This improves the results in [10], which showed that for $z > \frac{1}{2}$, $O(\frac{k}{\varepsilon^2})$ space suffices with a Count sketch. In both cases, occurrences of z cancel, so there is no dependency on z provided the distribution is skewed with $z > 1$. We can set the space based on k and ε without needing to know z exactly. Further, using a CM Sketch, one can simulate the sketch of [10] by computing $(count[j, 2i] - count[j, 2i-1])$ for all $1 \leq j \leq d, 1 \leq i \leq w/2$. The converse is not possible.

6 Second Frequency Moment Estimation

The second frequency moment, and the closely related L_2 norm, have been the focus of much study in the data stream community. The work of Alon, Matias and Szegedy [4] spurred interest in the data stream model of computation. One of their results was an efficient algorithm to compute the second frequency moment of a stream of values in space $O(\frac{1}{\varepsilon^2})$. As was observed by the authors of [19], the same algorithm also allowed the L_2 difference of two streams to be computed in a very general model. The algorithm can also be viewed as a streaming implementation of the Johnson-Lindenstrauss lemma [29] with limited randomness and bounded space. The lemma states that a set of vectors in Euclidean space can each be replaced by a vector in $O(\frac{1}{\varepsilon^2})$-dimensional space, and inter-vector distances are preserved up to a $(1 \pm \varepsilon)$ factor with high probability. This dependency on ε is essentially tight in terms of the dependency on ε for general distributions: a lower bound of $\Omega(\frac{1}{\varepsilon^2})$ has recently been shown [51]. This is problematic for applications that require a very fine quality approximation, say $\varepsilon = 1\%$ or 0.1% error, since the dependency on ε^{-2} means a high space cost. Here, we show how the CM sketch can be used to approximate this heavily studied quantity with strong guarantees of accuracy, and how, for skewed distributions, the $\Omega(\varepsilon^{-2})$ space bounds can be beaten for the first time.

6.1 Skewed Data

We describe the estimation procedure for the L_2 norm; to estimate the second frequency moment, we return the square of this value. When the distribution is skewed ($z > 1$), there are a few items with high frequency, and so a simple method to approximate the norm suffices. That is, we simply compute our estimate of the L_2 norm as

$$\min_j (\sum_k count[j,k]^2)^{1/2}$$

which is minimum of the L_2 norm of the rows of the sketch. We refer to this method as CM$^+$.

THEOREM 6.1. *This procedure estimates the L_2 norm of streams with Zipf skewness parameter > 1, with error bounded by $\varepsilon \|\boldsymbol{a}\|_2$ where $\varepsilon = O(w^{\frac{-(1+z)}{2}})$, with probability at least $1 - \delta = 1 - \frac{3}{4}^{-d}$.*

Proof. Let $m = w^{1/2}$. Then, with constant probability, in any row the largest m items fall in different buckets within the CM sketch. This follows from the pairwise independence of the hash functions used.

We compute the (squared) error in the jth estimator as $X_j = \sum_i count[j,i]^2 - \|\boldsymbol{a}\|_2^2$. Consider the expectation of this quantity when the above condition holds, that is, when the m largest counts are in m distinct buckets:

$$\mathsf{E}(X_j) = \sum_{i=1}^{U} a_i^2 + \sum_{i=1}^{U} \sum_{j=1, j\neq i}^{U} a_i a_j \mathsf{Pr}[h(i) = h(j)] - \|\boldsymbol{a}\|_2^2$$

$$\leq \|\boldsymbol{a}\|_2^2 + \frac{1}{w}(\sum_{i=1}^{U} \sum_{j=1,j\neq i}^{U} a_i a_j - \sum_{i=1}^{m} \sum_{j=1,j\neq i}^{m} a_i a_j) - \|\boldsymbol{a}\|_2^2$$

$$\leq \frac{\|\boldsymbol{a}\|_1^2}{w}(2\sum_{i=1}^{m} f_i \sum_{j=m+1}^{U} f_j + (\sum_{i=m+1}^{U} f_i)^2)$$

$$\leq 2\frac{\|\boldsymbol{a}\|_1^2}{w}(\sum_{i=1}^{U} f_i \sum_{j=m+1}^{U} f_j)$$

$$\leq \frac{2\|\boldsymbol{a}\|_1^2 c_z m^{1-z}}{w(z-1)} \leq \frac{2\|\boldsymbol{a}\|_2^2 c_z (2z-1)}{c_z^2(z-1)} w^{\frac{-(1+z)}{2}}$$

This makes use of the Facts 4.2 and 4.3 to bound the sum of the tail of the distribution and to relate the L_1 norm to the L_2 norm. Note that, since $a_i = \|\boldsymbol{a}\|_1 f_i$ and $\|\boldsymbol{a}\|_2^2 = \sum_i a_i^2$, we can write $\|\boldsymbol{a}\|_2^2 = \|\boldsymbol{a}\|_1^2 \sum_i f_i^2$. We can substitute this inequality, and then use the lower bound of Fact 4.3 to rewrite $\sum_i f_i^2$ in terms of z and c_z. We set the expected squared error equal to $\varepsilon\|\boldsymbol{a}\|_2^2/2$, which gives $w = O(\varepsilon^{\frac{-2}{1+z}})$. We treat the terms polynomial in z as effectively constant.

We then apply the Markov inequality, so with probability $\frac{3}{4}$, $X_j < 2\varepsilon\|\boldsymbol{a}\|_2^2$. This implies that $\|\boldsymbol{a}\|_2^2 \leq X_j \leq (1+\varepsilon)^2\|\boldsymbol{a}\|_2^2$. Taking the square root of all terms in this inequality bounds the L_2 norm of $\boldsymbol{a}$. For each row the probability of this failing to hold is no more than $\frac{3}{4}$: $\frac{1}{2}$ for the m items not falling in different counters, $\frac{1}{4}$ from the Markov inequality. Taking the minimum of these estimates amplifies the probability of success to $1 - \frac{3}{4}^d$. □

6.2 Moderate Skew

For the moderate skew ($z < 1$), and unskewed cases, we use the CM Sketch data structure to effectively simulate the sketch proposed by Alon Matias and Szegedy [4]. This shows the flexibility of the CM Sketch. In order for the results to be provable, we need to strengthen the hash functions used, from pairwise independent to 4-wise independent[2]. Apart from this change, the data structure is constructed and maintained in the same way as before.

Again, let $m = w^{1/2}$; it remains the case that the m largest items will not collide, although these contribute a smaller amount to the L_2 norm. Now, compute the estimate (denoted CM$^-$) of the L_2 norm for each row by taking the square root of

$$Y_j = \sum_{k=1}^{w/2}(count[j,2k] - count[j,2k-1])^2.$$

[2]In [4], the authors argue that in practice, pairwise or other hash functions will often suffice.

THEOREM 6.2. *With constant probability,*

$$(1-\varepsilon)\|\boldsymbol{a}\|_2^2 \le Y_j \le (1+\varepsilon)\|\boldsymbol{a}\|_2^2$$

for $\varepsilon = w^{\frac{-(1+2z)}{4}}$.

Proof. We will define some functions derived from the hash functions, h, in order to simplify the notation and clarify the analysis. We define $g_j(x) = +1$ if $h_j(x) \equiv 0 \mod 2$, and -1 otherwise. We also define $h'_j(x) = \lceil h_j(x)/2 \rceil$.

First, we will show that in expectation, $\mathsf{E}(Y_j) = \|\boldsymbol{a}\|_2^2$. Observe that

$$\begin{aligned}\mathsf{E}(Y_j) &= \textstyle\sum_{x,y} a_x a_y g_j(x) g_j(y) \mathsf{Pr}[h'_j(x) = h'_j(y)] \\ &= \textstyle\sum_x a_x^2 = \|\boldsymbol{a}\|_2^2\end{aligned}$$

using the pairwise independence of the function g (which follows from the pairwise independence of h). Secondly, we compute the variance of Y_j as

$$\begin{aligned}\mathsf{Var}(Y_j) &= \mathsf{E}(Y_j^2) - \mathsf{E}(Y_j)^2 \\ &= (\textstyle\sum_{i=1}^{w/2} (\sum_{x, h'_j(x)=i} a_x g_j(x))^2)^2 - \|\boldsymbol{a}\|^4 \\ &\le \textstyle\sum_{v,x,y,z} 4 g_j(v) g_j(x) g_j(y) g_j(z) a_v a_x a_y a_z \\ &\quad * \mathsf{Pr}[h'_j(v) = h'_j(x) = h'_j(y) = h'_j(z)] \\ &= 4 \textstyle\sum_{x=1}^{U} \sum_{y=1, y\ne x}^{U} a_x^2 a_y^2 \mathsf{Pr}[h'_j(x) = h'_j(y)] \\ &= \tfrac{4}{w} \textstyle\sum_{x=1}^{U} \sum_{y=1, y\ne x}^{U} a_x^2 a_y^2\end{aligned}$$

This uses the 4-wise independence of the function h to imply 4-wise independence of g, and hence to show that products of 4 or fewer independent terms in g have expectation zero.

We again argue that, with probability at least $\frac{1}{2}$, the $m = w^{1/2}$ largest counts fall into different buckets. Consider the distribution of counts in the CM sketch only for such settings where this event occurs. For such distributions, then $\mathsf{Var}(Y_j)$ is bounded as:

$$\begin{aligned}\mathsf{Var}(Y_j) &\le \tfrac{4\|\boldsymbol{a}\|_1^4}{w} (2 \textstyle\sum_{i=1}^{m} f_i^2 \sum_{j=m+1}^{U} f_j^2 \\ &\quad + (\textstyle\sum_{i=m+1}^{U} f_i^2)^2) \\ &\le \tfrac{4\|\boldsymbol{a}\|_1^2}{w} (2 \textstyle\sum_{i=1}^{U} f_i^2 \sum_{j=m+1}^{U} f_j^2) \\ &\le 8\|\boldsymbol{a}\|_2^4 m^{1-2z}/w = 8\|\boldsymbol{a}\|_2^4 w^{\frac{-1-2z}{2}}\end{aligned}$$

Setting this equal to $\varepsilon^2\|\boldsymbol{a}\|_2^4$ lets us apply the Chebyshev bound. This shows that $\mathsf{Pr}[|Y_j - \|\boldsymbol{a}\|_2^2| > 2\varepsilon\|\boldsymbol{a}\|_2^2] < \frac{1}{4}$ provided we have $\varepsilon^2 \ge 8w^{\frac{-(1+2z)}{2}}$. We can take the median of $O(\ln\frac{1}{\delta})$ independent repetitions of the estimator Y_j and apply Chernoff bounds in usual way to amplify this constant probability of success to $1-\delta$. The space required is $O(\varepsilon^{\frac{-4}{1+2z}} \ln\frac{1}{\delta})$ for $z > \frac{1}{2}$. □

6.3 Light Skew Case and Summary

For the case where $z \le \frac{1}{2}$, we observe that by simply taking the variance of the CM$^-$ estimator over all distributions, then it is directly bounded as $\mathsf{Var}(Y_j) \le 8\|\boldsymbol{a}\|_2^4/w$. Following the Chebyshev and Chernoff arguments results in space bounds of $O(\frac{1}{\varepsilon^2})$. This matches the space requirements for the previously best known algorithms for L_2 estimation of [4, 13, 49] up to small constant factors. Observe that the update time is the same as the usual cost for updating a CM Sketch, which is $O(d) = O(\ln\frac{1}{\delta})$. Here we give much improved dependency on ε on space used for skewed distributions, as summarized in the table below.

Value of z	$z \le \frac{1}{2}$	$\frac{1}{2} < z \le 1$	$1 < z$
Space required	$O(\varepsilon^{-2})$	$O(\varepsilon^{\frac{-4}{1+2z}})$	$O(\varepsilon^{\frac{-2}{1+z}})$

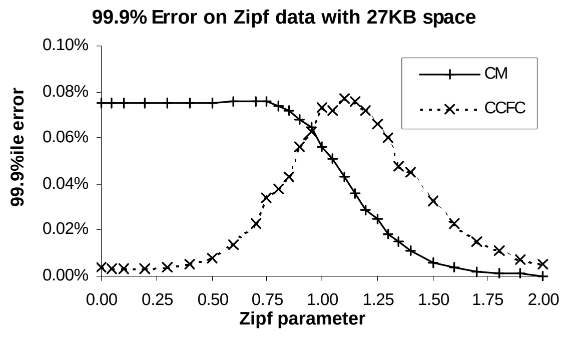
7 Experimental Study

We carried out an extensive experimental analysis of the Count Min sketch for point estimation and L_2 estimation. We made use of the public implementations of the data structure available from `http://www.cs.rutgers.edu/~muthu/massdal-code-index.html` as well as the Count sketch [10] for comparison. The Count sketch can also be used to answer point queries, and has a similar structure to the Count-Min sketch, being based around an array of counters. So in all experiments, the two methods were given exactly the same amount of space, in each case arranged as an array of counters with the same dimensions. This should give a fair comparison between the two methods. We refer to the Count-Min sketch as “CM”, and the Count sketch as “CCFC” (after the initials of its creators) for brevity. We considered synthetic datasets generated from Zipf distributions with known values of z, so that we could compare the behavior of the algorithm with that predicted by our analysis. We also considered various real data sets, two data sets in each category, text and network data.

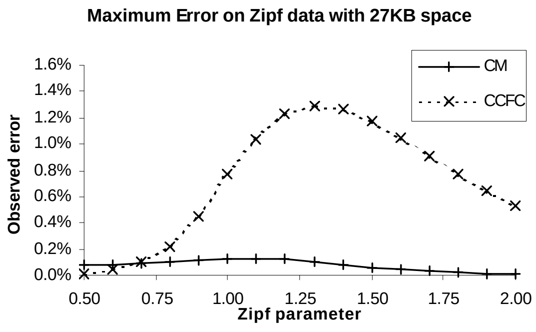
7.1 Synthetic Data

We made our synthetic data sets by using standard routines to draw values from a Zipf distribution with specified parameter z. Each experiment consisted of drawing 10^7 items from a domain of size $n = 10^6$ and computing the L_2 norm and all point queries over this domain. In evaluating the quality of our algorithms, we computed the exact solutions to all these queries, and so could compute the error in each result. For L_2 norms, we computed the fractional error as the difference between the estimated and actual value, scaled by the actual value. For simplicity, we worked with $F_2 = L_2^2$. For point queries, we computed the difference between the actual value and the estimate, and scaled by the number of items. We computed the maximum error observed, and the 99.9th percentile of the error (that is, sort the observed errors, and take the one whose rank is $\frac{999}{1000}N$). Since our algorithms give guaranteed bounds with a small probability of failure, this should test how well these bounds are met.

The first results are shown in Figure 1. These show the effect of fixing the space s for algorithms, but varying the skewness parameter of the input. Our theory predicts that the performance of the Count-Min sketch should be $\varepsilon \propto 1/s$ for $z < 1$, and $\varepsilon \propto 1/s^z$ for $z > 1$. We observe that this seems to be borne out in Figure 1(a): the error is roughly flat

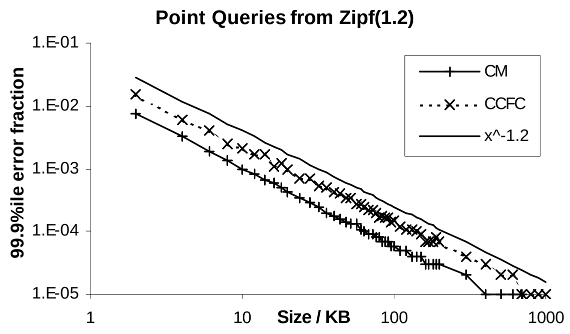


(a)

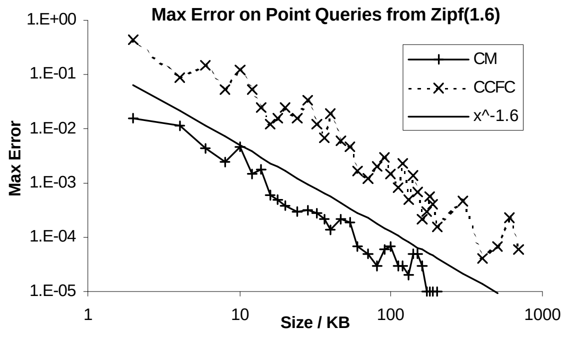


(b)

Figure 1: Testing point estimation on synthetic data



(a)



(b)

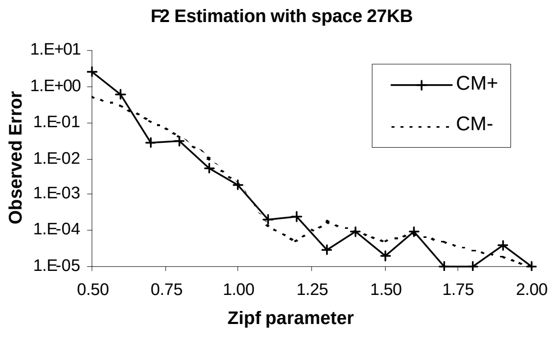
Figure 2: Testing space dependency of point estimation on synthetic data

for $0.5 < z < 1$, and then falls off smoothly for larger z. We see that for $z > 1$ then the observed error is better for CM than for CCFC, justifying our analysis of the performance of these algorithms for skewed data sets. For distributions with skew $z \geq 2$, the observed error is sufficiently small to be negligible. A similar pattern of behavior is seen when we take the maximum observed error, in Figure 1(b). The main observation is that for skewed data, the largest error from the CCFC approach can become very high compared to that of CM, which is not much greater than in the previous case.[3]

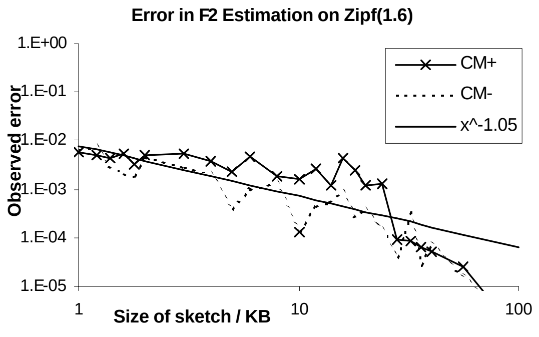
Our theory predicts that, as space s increases, the error ε should decrease as s^{-z}. We show this to be the case in Figure 2. We plot the observed error when we fix the Zipf parameter, and increase the space for the sketch from 1KB to 1MB. Plotting observed error vs. space on a log-log plot should reveal a straight line with slope equal to $-z$. This is seen clearly in Figure 2(a), where we have plotted a line $y \propto x^{-1.2}$ for comparison. Note that this is a logarithmic scale plot, so the separation between the two lines is quite significant: CCFC consistently has about twice as much error. It appears to show a similar $x^{-1.2}$ dependency on size. Although the maximum error is much more variable, the same behavior occurs for $z = 1.6$ (Figure 2(b)), where CCFC has on average 10 times the error of CM, an order of magnitude worse. Several data mining problems need to manipulate item counts by summing and subtracting estimated values, so often this very fine accuracy is required, hence the need to get as good quality estimates as possible in small space.

For F_2 estimation, the results are less clearcut. We have two methods to use the Count-Min sketch in order to estimate the second frequency moment. The first, CM$^+$,

[3]We do not know why the CCFC algorithm appears to have a "bell-curve"-like behavior as z increases. This may be of interest for future analysis.

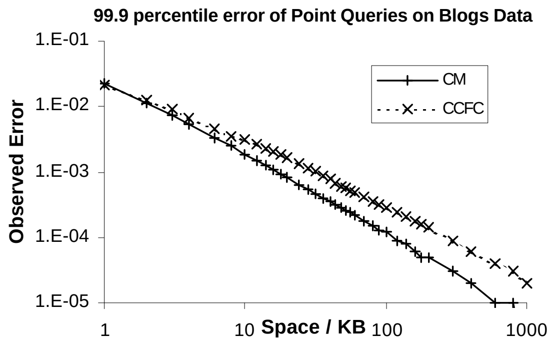


(a)

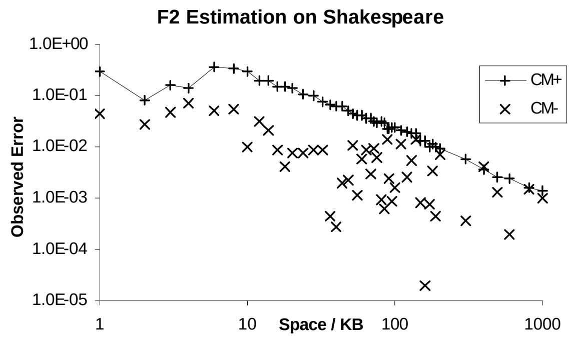


(b)

Figure 3: Testing space L_2 estimation on synthetic data



(a)



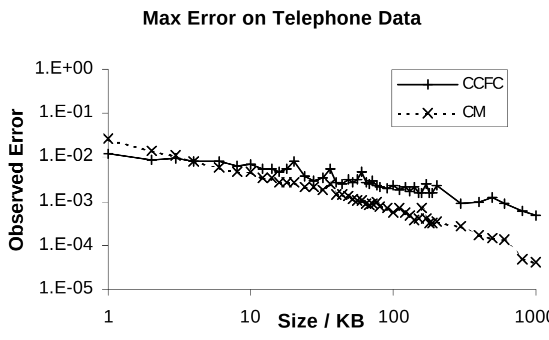
(b)

Figure 4: Results of experiments on text data

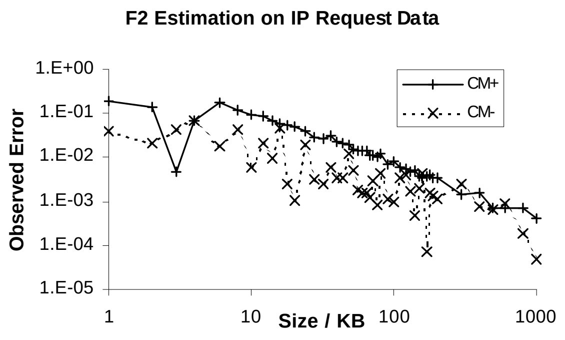
which has the best analytic results for $z > 1$, is to sum the squares of the counters. The second, CM$^-$, sums the squares of the differences of adjacent counters. In our experiments on synthetic data, illustrated in Figure 3, we were not able to observe a clear difference between the two approaches. As we increased z while keeping the space fixed, we saw that both methods seem to give about the same error. In Figure 3 (a), over the different values of z, CM$^+$ gets lower error more often than CM$^-$, but there is no clear trend. Figure 3 (b) shows the effect of increasing the size of the sketch for data with $z = 1.6$. Our theory predicts that the error of CM$^-$ should behave as $s^{-\frac{1+2z}{4}} = s^{-1.05}$, and CM$^+$ as $s^{-\frac{1+z}{2}} = s^{-1.3}$. We have plotted the first of these on the same graph, since on this data set we can see this behavior for CM$^-$. The results for CM$^+$ are much less clear here, however when we examine real data sets we shall see the algorithms performing very closely in line with their predicted behavior.

7.2 Text Data

Zipf's law was first proposed in the context of linguistics, based on the observation that the frequency of the ith most frequent word in written text seemed to be roughly proportional to $1/i$ [53]. So it is fitting that we test our methods on mining textual data. We considered two data sources of seemingly very different nature. First, we used the complete plays of Shakespeare. This consists of 5MB of data, totaling approximately 1 million words. As a data source, it is quite 'clean', since words are spelled consistently throughout, and has been checked by many editors. Our second source of data consisted of a large amount text harvested from weblogs ("blogs"), totaling 1.5GB. This totaled over 100 million words from a large number of different authors, written in colloquial English (and some other languages mixed in), with no editing, in inconsistent styles and many errors left uncorrected. We did not attempt to clean this data, but ran our algorithms on it directly.



(a)



(b)

Figure 5: Results of experiments on network data

We show some of the results in Figure 4. We do not show all results for space reasons, but they are similar to those we present. On the log-log plot in Figure 4 (a) we can see a very clear linear gradient with slope approximately -1.2. This suggests that the Blog data is well-modeled by a Zipf distribution with parameter 1.2. We took the word frequencies from this data, and plotted those on a log-log chart, then computed the line of best fit; its slope was indeed approximately 1.2.[4] For the Shakespeare dataset, we measured z as 1.2—1.3, implying that Shakespeare had similar relative frequencies of word usage as a Blog writer. In both cases, we see significant accuracy gains using CM Sketch over CCFC.

The linear behavior of the CM^+ estimation in Figure 4 (b) is quite striking. For sketch sizes 10KB—1MB, we measured a dependency of ε as $s^{-1.15}$. This implies a corresponding value of $z = 1.3$. The same trend is not obvious for the CM^- approach, but a best fit line gives the dependency $\varepsilon \propto s^{-0.85}$, which corresponds to $z = 1.2$. Recall that in both cases, we use the same sketch as the basis of the both estimation procedures (as well as for point queries). These results show us that using the CM^- estimation technique (sum of squares of differences) gives better results than the CM^+ approach (sum of squares). If the data is very skewed, or very fine accuracy is required, then CM^+ should be used, since asymptotically it has better bounds, but for this kind of data the CM^- method is preferable. The important feature is that we can make the sketch oblivious to the nature of the data, and only at query time decide which estimation technique to use, based on the observed skew.

[4] In order to get a good fit of real data to a Zipf distribution, one typically has to drop the first few readings, and not fit the entire tail. Based on different sections of this chart, we measured Zipf parameters in the range 1.15 to 1.30, and so we conclude that 1.2 is within the bounds of uncertainty.

7.3 Network Data

We considered two types of data drawn from communication networks: a data set of 1.9 million phone calls, where we tracked the called exchange (a range of 1 million values); and a data set of Internet requests to 32-bit IP addresses, taken from the Internet Traffic Archive [38], LBL-CONN7 [46], totaling 800,000 requests. The maximum error on the phone call data is plotted in Figure 5 (a). Although it is a little fuzzier than the corresponding 99.9% error plot, the linear dependency on the log-log plot can be easily seen. The slope of the CM line is -1.16, predicting a skewness parameter of 1.16, while the slope of the CCFC line is around -0.8. Again, there is an order of magnitude improvement in the accuracy of CM over CCFC. For the Internet data, the error in point queries implies a skew of $z = 1.3$. This means that the slope for F_2 estimation should be 1.15, which is indeed what we measure on Figure 5 (b) for sketches between 10KB and 1MB using CM^+ for estimation. The slope of the CM^- line is less steep, about -0.9 as predicted, although again the observed error is less throughout most of the region of interest.

7.4 Timing Results

Since the update procedure is essentially the same for every update, the time cost is not much affected by the nature of the data. We conducted experiments on 1GHz and 2.4GHz processor machines, and observed similar update performance on each (since the algorithm is essentially bound by cache/memory access times), of about 2–3 million updates per second. By comparison, the implementation of the CCFC Count Sketch achieves a somewhat slower rate (40–50% slower), since it requires additional computation of a second hash function for ever update. Greater speed can be achieved by taking advantage of the natural parallelism inherent in sketch data structures.

8 Conclusions

We have defined the problem of summarizing and mining data streams when these streams exhibit a skewed distribution. We have given practical algorithms for key post-hoc analysis problems with strong theoretical bounds of $o(1/\varepsilon)$ and $o(1/\varepsilon^2)$ where previously known results that did not exploit skew used space $\Omega(1/\varepsilon)$ and $\Omega(1/\varepsilon^2)$ respectively. In experiments, we have shown our CM sketch data structure to be a practical and flexible summary: not only does it outperform other methods for point queries and give accurate estimates for L_2 estimation, but it does this based on a simple update procedure. This approach can be employed without *a priori* knowledge of the distribution or skewness of the data: given fixed space, we can then bound the approximation quality based on the observed skew.

The two queries that we considered are fundamental to top-k items, change detection, approximate quantiles, anomaly detection and so on. Many other summarization and mining tasks can also benefit from the insight that data is rarely uniform, and realistic data is frequently highly skewed. For example, we remark that our methods in this paper will give estimates for *inner-product queries* between data streams as well in a straightforward way as an extension of [13]. This has applications to join size estimation in databases [3], to principal component analysis [31] and sparse correlation matrix estimation [32], but we do not elaborate further on this here. Likewise, the fact that skew is frequently seen at multiple levels of aggregation [33] means that our analysis can be immediately applied to *hierarchical computations*, such as computing range sums, estimating quantiles and so on (see [13] for these computations using CM sketch). With appropriate analysis and testing, methods that capitalize on data skew could improve our understanding of existing algorithms, inspire new methods, and move some tasks previously thought unachievable into the practical.

Acknowledgments We thank Yinmeng Zhang for some useful discussions, and the referees for their suggestions.

References

[1] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R. Yan, and S. Zdonik. Aurora: a data stream management system. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 666–666, 2003.

[2] L. Adamic. Zipf, power-law, pareto - a ranking tutorial. `http://www.hpl.hp.com/research/idl/papers/ranking/`, 2000.

[3] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proceedings of the Eighteenth ACM Symposium on Principles of Database Systems*, pages 10–20, 1999.

[4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996. Journal version in *Journal of Computer and System Sciences*, 58:137–147, 1999.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of ACM Principles of Database Systems*, pages 1–16, 2002.

[6] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

[7] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace. Discovering hidden groups in communication networks. In *2nd NSF/NIJ Symposium on Intelligence and Security Informatics*, pages 126–137, 2004.

[8] A. Bestavros, M. Crovella, and T. Taqqu. *Heavy-Tailed Probability Distributions in the World Wide Web*, pages 3–25. Birkhäuser, 1999.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM*, pages 126–134, 1999.

[10] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Procedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.

[11] G. Cormode, F. Korn, S. Muthukrishnan, T. Johnson, O. Spatscheck, and D. Srivastava. Holistic UDAFs at streaming speeds. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 35–46, 2004.

[12] G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proceedings of ACM Principles of Database Systems*, pages 296–306, 2003.

[13] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *Latin American Informatics*, pages 29–38, 2004.

[14] G. Cormode and S. Muthukrishnan. What's new: Finding significant differences in network data streams. In *Proceedings of IEEE Infocom*, 2004.

[15] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.

[16] A. Dobra, M. Garofalakis, J. E. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM Sigmod International Conference on Management of Data*, pages 61–72, 2002.

[17] C. Estan and G. Varghese. Data streaming in computer networks. In *Proceedings of Workshop on Management and Processing of Data Streams*, `http://www.research.att.com/conf/mpds2003/schedule/estanV.ps`, 2003.

[18] W. Fei, P. S. Yu, and H. Wang. Mining extremely skewed trading anomalies. In *International Conference on Extending Database Technology*, pages 801–810, 2004.

[19] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L_1-difference algorithm for massive data

streams. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 501–511, 1999.

[20] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2002.

[21] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 389–398, 2002.

[22] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of the International Conference on Very Large Data Bases*, pages 79–88, 2001. Journal version in *IEEE Transactions on Knowledge and Data Engineering*, 15(3):541–554, 2003.

[23] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proceedings of the International Conference on Very Large Data Bases*, pages 454–465, 2002.

[24] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 32(2):5–14, June 2003.

[25] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical Report SRC 1998-011, DEC Systems Research Centre, 1998.

[26] The himalaya project. `http://www.cs.cornell.edu/database/himalaya/Himalaya.htm`.

[27] B. Huberman, P. Pirolli, J. Pitkow, and R. Lukose. Strong regularities in world wide web surfing. *Science*, pages 95–97, April 1998.

[28] IBM Research — stream data mining. `http://www.research.ibm.com/compsci/project_spotlight/kdd/`.

[29] W.B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[30] NASA jet propulsion laboratory. `http://www7.nationalacademies.org/bms/BravermannasPDF.pdf`.

[31] H. Kargupta and V. Puttagunta. An efficient randomized algorithm for distributed principal component analysis for heterogenous data. In *Workshop on high performance data mining at SIAM Intl Conf on Data mining*, 2004.

[32] H. Kargupta and V. Puttagunta. Onboard vehicle data stream monitoring and fast computation of sparse correlation matrices. In *Workshop on data mining in resource constrained environments at SIAM Intl Conf on Data mining*, 2004.

[33] E. Kohler, J. Li, V. Paxson, and S. Shenker. Observed structure of addresses in IP traffic. In *ACM SIGCOMM Internet Measurement Workshop*, pages 253–266, 2002.

[34] N. Koudas and D. Srivastava. Data stream query processing: A tutorial. In *Proceedings of the International Conference on Very Large Data Bases*, page 1149, 2003.

[35] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation and applications. In *Proceedings of the ACM SIGCOMM conference on Internet measurement*, pages 234–247, 2003.

[36] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proceedings of the WWW Conference*, pages 568–576, 2003.

[37] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[38] Internet traffic archive. `http://ita.ee.lbl.gov/`.

[39] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of 18th International Conference on Data Engineering*, pages 555–566, 2002.

[40] D. Madigan. DIMACS working group on monitoring message streams. `http://stat.rutgers.edu/~madigan/mms/`, 2003.

[41] MAIDS : Mining alarming incidents in data streams. `http://maids.ncsa.uiuc.edu/`.

[42] G.S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the International Conference on Very Large Data Bases*, pages 346–357, 2002.

[43] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[44] S. Muthukrishnan. Data streams: Algorithms and applications. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[45] Ohio State CSE — algorithms for mining data streams. `http://www.cse.ohio-state.edu/~agrawal/Research_new/mining.htm`.

[46] V. Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE ACM Transactions on Networking*, 2(4):316–336, 1994.

[47] S. Redner. How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B*, pages 131–134, 1998.

[48] Stanford stream data manager. `http://www-db.stanford.edu/stream/sqr`.

[49] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 615–624, 2004.

[50] H. Wang. Bibliography on mining data streams. `http://wis.cs.ucla.edu/~hxwang/stream/bib.html`.

[51] D. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 167–175, 2004.

[52] Y. Zhu and D. Shasha. StatStream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the International Conference on Very Large Data Bases*, pages 358–369, 2002.

[53] G. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison Wesley, 1949.