Summarizing Data using Bottom-k Sketches

Edith Cohen
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
edith@research.att.com

Haim Kaplan
School of Computer Science
Tel Aviv University
Tel Aviv, Israel
haimk@cs.tau.ac.il

ABSTRACT

A *Bottom-k sketch* is a summary of a set of items with nonnegative weights that supports approximate query processing. A sketch is obtained by associating with each item in a ground set an independent random rank drawn from a probability distribution that depends on the weight of the item and including the k items with smallest rank value.

Bottom-k sketches are an alternative to k-mins sketches [9], which consist of the k minimum ranked items in k independent rank assignments, and of min-hash [5] sketches, where hash functions replace random rank assignments. Sketches support approximate aggregations, including weight and selectivity of a subpopulation. Coordinated sketches of multiple subsets over the same ground set support subset-relation queries such as Jaccard similarity or the weight of the union. All-distances sketches are applicable for datasets where items lie in some metric space such as data streams (time) or networks. These sketches compactly encode the respective plain sketches of all neighborhoods of a location. These sketches support queries posed over time windows or neighborhoods and time/spatially decaying aggregates.

An important advantage of bottom-k sketches, established in a line of recent work, is much tighter estimators for several basic aggregates. To materialize this benefit, we must adapt traditional k-mins applications to use bottom-k sketches. We propose all-distances bottom-k sketches and develop and analyze data structures that incrementally construct bottom-k sketches and all-distances bottom-k sketches.

Another advantage of bottom-k sketches is that when the data is represented explicitly, they can be obtained much more efficiently than k-mins sketches. We show that k-mins sketches can be derived from respective bottom-k sketches, which enables the use of bottom-k sketches with off-the-shelf k-mins estimators. (In fact, we obtain tighter estimators since each bottom-k sketch is a distribution over k-mins sketches).

Categories and Subject Descriptors: E.2 Data Storage Representations; G.3: probabilistic algorithms; E.1 Data Structures

General Terms: Algorithms, Measurement, Performance, Theory

Keywords: all-distances sketches, data streams, bottom-k sketches

1. INTRODUCTION

Sketching or sampling is an extremely useful tool for storage and queries on massive data sets. Sketches allow us to process approximate queries on the original data sets while occupying a fraction of the storage space required for the full data set and using a fraction of the computation resources required for the exact answer. The value of a sketching method depends on the efficiency of its implementation, its versatility in terms of the operations supported, and the quality of the estimates obtained.

Bottom-k and *k-mins* sketches are summaries of a set of items with positive weights. k-mins sketches (The *min-rank* method [9]) are obtained by assigning independent random ranks to items where the distribution used for each item depends on the weight of the item. We retain the minimum rank of an item in the set. This is repeated with k independent rank assignments for some integer $k \geq 1$ and we obtain a k-vector of independent minimum ranks and k independent weighted samples. *Bottom-k* sketches are an emerging alternative to k-mins sketches. Bottom-k sketches are constructed using a single rank assignment. The bottom-k sketch of a subset contains the k items with smallest ranks in the subset. Bottom-k sketches were mentioned, without analysis, in [9, 22].

The sketch supports approximate query processing over the original data set and subpopulations of this dataset. Basic aggregations include the weight of the set or the selectivity of a subpopulation (subset) of the set and derived aggregations include approximate quantiles, average weight, and variance and higher moments [10]. The sketch of a set is a weighted random sample. When used with exponentially distributed ranks, bottom-k sketches are a weighted sample without replacement (WS-sketches) whereas k-mins sketches are a weighted sample with replacement (WSR-sketches).

In applications where there are multiple subsets that are defined over the same ground set of items, a sketch is produced for each subset. The sketches of different subsets are "coordinated," sharing the same rank assignments to the items of the ground set, and support queries over subset relations, such as the weight of the union or intersection, their weight ratio, and resemblance or Jaccard similarity coefficient. A useful property of "coordinated" sketches is that the sketch of a union can be computed from the sketches of the subsets. Therefore, given sketches of subsets, we can perform aggregations on unions of subsets.

Example of an application with multiple subsets is when items are associated with nodes of a directed graph and we compute k-mins sketches for the reachability set of each node. These sketches can be computed in $\tilde{O}(km)$ time (and storage) whereas an explicit representation of the subsets requires $O(mn)$ time [9]. Applications include maintaining a sketch of influencing events for each process in a computer system [15], when a process A affects process B, the new sketch of B becomes the sketch of the union; and using the property that the sketches reduce the approximate sum problem to that of finding a minimum, k-mins sketches were used for aggregations on gossip networks [21].

Other applications with multiple subsets where sketches support fast computation of subset relations are near-duplicate detection for Web pages [5] (a sketch is produced for each Web page), study of similar Web sites [2], mining of association rules [22] from market basket data, and eliminating redundant network traffic [23]. In these applications, a variant termed *min-hash* sketches substitutes random rank assignments with random hash functions (families of min-wise independent hash functions or ϵ-min-wise functions [5, 6]). With random hash functions, the rank assignment of an item depends on the *item identifier*, and it has the property that all copies of the same item across different subsets obtain the same rank, without additional book keeping or coordination between all occurrences of each item. This allows for efficient aggregations over *distinct occurrences* (see [19]) and supports subset-relation queries.

Bottom-k sketches encode more information than k-mins sketches. (Intuitively, sampling without replacement is more informative than sampling with replacement.) A line of recent work showed that bottom-k sketches are superior to k-mins sketches in terms of estimate quality. Estimators for subpopulation weight using priority ranks (PRI-sketches) were provided in [1, 24] and estimators for general families of rank functions were provided in [11, 12]. The improvement in estimate quality is significant on weight distributions and values of k, such that items are likely to be sampled multiple times in a k-sample drawn with replacement, such as skewed Zipf-like distributions that often arise in practice. For subset relations such as the weight of the intersection or union, bottom-k sketches improve over k-mins sketches even when weights are uniform [11, 12]: Carefully designed estimators are applied to the combined bottom-k sketches, which reveal more members of the union and intersection than two corresponding k-mins sketches.

Our contributions

We facilitate the use of bottom-k sketches by developing and analyzing data structures that construct these sketches. Our results allow applications that use k-mins sketches to use the superior bottom-k sketches. An inherent difference we had to tackle is that k-mins sketches are obtained using k independent rank functions, which allows for k independent copies of the same simple data structure to be used whereas bottom-k entries are dependent.

Sketches are constructed incrementally as items are processed. The sketch is manipulated through two basic operations: A *test* operation which tests if the sketch has to be updated, and an *update* operation which inserts the new item if the sketch indeed has to be updated. We make this distinction since test operations can be performed much more efficiently than update operations. The number of update operations depends on the order in which items are processed and on the weight distribution of the data. The number of test operations is typically larger than the number of updates. The extent in which it is larger, however, highly depends on the application.

We distinguish between applications with *explicit representation* [3, 2, 22, 23] or *implicit representation* [9, 13, 15] of the data. In applications with an explicit representation, item-subset pairs are provided explicitly. The dataset could be distributed, presented as a data stream, or in external memory, but the pairs are explicitly provided and are all processed to produce the sketches. In applications with implicit representation, the subsets are specified as neighborhoods in a graph or some metric space. With explicit representation, the number of test operations is much larger than the number of update operations. In Section 3 we analyze the number of test and update operations and how it depends on the way the data is presented and on the distribution of the item weights.

All-distances sketches are a generalization of plain sketches that are used when the underlying dataset has items associated with locations in some metric space, and subsets are specified by neighborhoods of a location. All-distances k-mins sketches were used for data streams (where aggregation is over windows of elapsed time to the present time) [14], the Euclidean plane (where we are presented with a query point and distance) [13, 20], a graph (the query is a node and distance) [9], or distributed "spatial aggregation" over a network [9, 13]. An all-distances sketch is a compact encoding of the plain sketches of all neighborhoods of a certain location q. For a given distance d, the sketch for the d-neighborhood of the location can be constructed from the all-distances sketch. All-distances sketches also support time-decaying and spatially-decaying aggregates using arbitrary decay functions [14, 13]. In Section 4 we define bottom-k all-distances sketches and present efficient data structures for maintaining both all-distances k-mins sketches and all-distances bottom-k sketches. We analyze the number of operations required to construct all-distances sketches under different arrival orders of the items.

In Section 6 we provide a method to derive WSR-sketches (k-mins with exponential ranks) from WS-sketches (bottom-k with exponential ranks). This *mimicking process* provides a general method of applying estimators designed for WSR-sketches to WS-sketches. This process enables us to use bottom-k sketches in applications (such as those with explicit representation of the data) where they can be obtained much more efficiently than k-mins sketches and use readily available WSR-sketches estimators. In fact, since each WS-sketch corresponds to a distribution over WSR-sketches, we obtain estimators with smaller variance than the underlying WSR-sketches estimators. This reduction also shows that WS-sketches are strictly superior to WSR-sketches. We provide examples of applications of the mimicking process.

2. PRELIMINARIES

Let I be a ground set of items, where item $i \in I$ has weight $w(i) \geq 0$. A *rank assignment* maps each item i to a random rank $r(i)$. The ranks of items are drawn independently using a family of distributions $\mathbf{f_w}$ ($w \geq 0$), where the rank of an item with weight $w(i)$ is drawn according to $\mathbf{f_{w(i)}}$.

We use random rank assignments to obtain *sketches* of subsets as follows. For a subset J of items and a rank assignment r we define $B_1(r, J) = \arg\min_{j \in J} r(j)$, to be the item in J with smallest rank according to r. For $i \in \{1, \ldots, |J|\}$, we define $B_i(r, J)$ to be the item in J with ith smallest rank according to r and $r_i(J) \equiv r(B_i(r, J))$ to be the ith smallest rank value in J according to r.

Definition 2.1. k-mins sketches *are produced from k independent rank assignments, $r^{(1)}, \ldots, r^{(k)}$. The k-mins sketch of a subset J is the k-vector $(r_1^{(1)}(J), r_1^{(2)}(J), \ldots, r_1^{(k)}(J))$.*

To support some queries, we may need to include with each entry an identifier or some other attributes such as the weight of the items $B_1(r^{(j)}, J)$ $(j = 1, \ldots, k)$.

Definition 2.2. Bottom-k sketches *are produced from a single rank assignment r. The bottom-k sketch $s(r, J)$ of the subset J is a list of entries $(r_i(J), w(B_i(r, J)))$ for $i = 1, \ldots, k$. The list is ordered by rank, from smallest to largest.*

The bottom-k sketch of a subset is therefore a list with up to k entries. The size of the list is the minimum of k and the number of items in the subset. For a single item i (a subset of size 1), the bottom-k sketch is a list with a single entry $(r_1(J), w(B_1(r,J)))$. To support queries, in addition to the weight, entries in the sketch may include an identifier and attribute values of items $B_i(r,J)$ ($i = 1,\dots,k$).

Bottom-k and k-mins sketches have the following useful property: The sketch of a union of two sets can be generated from the sketches of the two sets. Let J and H be two subsets. For any rank assignment r, $r_1(J \cup H) = \min\{r_1(J), r_1(H)\}$. Therefore, for k-mins sketches we have $(r_1^{(1)}(J \cup H), \dots, r_1^{(k)}(J \cup H)) = (\min\{r_1^{(1)}(J), r_1^{(1)}(H)\}, \dots, \min\{r_1^{(k)}(J), r_1^{(k)}(H)\})$. For bottom-$k$ sketches, the k smallest ranks in the union $J \cup H$ are contained in the union of the sets of the k-smallest ranks in each of J and H. That is, $s(r, J \cup H) \subset s(r,J) \cup s(r,H)$. Therefore, the bottom-$k$ sketch of $J \cup H$ can be computed by taking the entries with k smallest ranks in the combined sketches of J and H.

To support sketch-based set operations and queries, we need to store the rank values of items. To perform sketch-based queries on a single subset, however, we do not need all rank values. With bottom-k sketches, it is sufficient to store the $(k+1)$st smallest rank value, r_{k+1}: We (re)draw random rank values for each item i in the sketch using $\mathbf{f_{w(i)}}$ conditioned on the rank being smaller than r_{k+1}. This is just like (re)drawing a random bottom-k sketch from the probability subspace where the minimum rank of items not in the sketch is equal to r_{k+1} and all items in the sketch have ranks smaller than r_{k+1}.

Beyond reduced storage, this observation often enables us to obtain tighter estimators. The unbiased *rank conditioning* estimator for subpopulation weight [11, 12] is applied to the value r_{k+1} and the weights of the items in the (unordered) sketch. In some cases, however, it is easier to derive estimator that is applied to the ordered sketch with rank values (the mimicking process in Section 6 is applied to an ordered bottom-k sketch). In this case, instead of applying an estimator to the original sketch and rank values, we take its expectation over re-drawn sketches or its average over multiple draws (if the expectation is hard to compute). This results in an estimator with at most the same variance and often smaller variance. Correctness follows from a basic property of variances:

Lemma 2.3. *Let a_1 and a_2 be two random variables over Ω. Suppose there is a partition of Ω such that the value of a_2 on each part is equal to the expectation of a_1 on that part. Then* $\text{VAR}(a_2) \le \text{VAR}(a_1)$.

The choice of which family of random rank functions to use matters only when items are weighted. Otherwise, sketches produced using one rank function can be transformed to any other rank function.[1]

WS-sketches and WSR-sketches. A convenient choice for the rank function $\mathbf{f_w}$ is an exponential distribution with parameter w [9]. The density function of this distribution is $\mathbf{f_w(x) = we^{-wx}}$, and its cumulative distribution function is $\mathbf{F_w(x) = 1 - e^{-wx}}$. We refer to k-mins sketches with these ranks as WSR-sketches and to bottom-k sketches with these ranks as WS-sketches.

The minimum rank $r_1(J)$ of an item in a subset $J \subset I$ is exponentially distributed with parameter $w(J) = \sum_{i \in J} w(i)$. This follows from the fact that the minimum of random variables each drawn from an exponential distribution is also an exponentially distributed random variable with parameter equal to the sum of the parameters of these distributions. The item with the minimum rank $B_1(r,J)$ is a *weighted random sample* from J: The probability that an item $i \in J$ is the minimum rank item is $w(i)/w(J)$.

Therefore we can conclude that a WSR-sketch of size k of a subset J is a weighted random sample of size k, drawn **with replacement** from J (hence the term WSR-sketches). The ranks of these items is a set of k independent samples from an exponential distribution with parameter $w(J)$. Hence, if the weight $w(J)$ is provided and we do not use subset-relation queries rank values are redundant. If $w(J)$ is not provided, the rank values can be used in unbiased estimators for both $w(J)$ and the inverse weight $1/w(J)$ [9].[2]

On the other hand, the items in a WS-sketch are samples drawn **without replacement** from J:

Lemma 2.4. *A* WS-*sketch of size k of a subset J is a sample of size k drawn without replacement from J.*

PROOF. The probability that item $i \in J$ is $B_1(r,J)$ is $w(i)/w(J)$. Conditioned on the bottom-j ranked items in J being $i_1,\dots,i_j$, $B_{j+1}(r,J)$ is $i \in J \setminus \{i_1,\dots,i_j\}$ with probability $w(i)/(w(J) - \sum_{h=1}^{j} w(i_h))$. □

If the weight $w(J)$ is provided and we do not use the sketches for subset-relation queries it suffices to store the unordered set of items in $s(r,J)$. This information allows us to draw at random a bottom-k sketch from the probability subspace that contains all sketches where the set of the bottom-k ranked items is $s(r,J)$.

PRI-sketches. With priority ranks [18, 1] the rank value of an item with weight w is selected uniformly at random from $[0, 1/w]$. This is the equivalent to choosing rank value r/w, where $r \in U[0,1]$ is selected from the uniform distribution on the interval $[0,1]$. It is well known that if $r \in U[0,1]$ then $-\ln(r)/w$ is an exponential random variable with parameter w. Therefore exponential ranks correspond to using rank values $-\ln r/w$ where $r \in U[0,1]$.

Choice of a rank function. The appeal of PRI-sketches is estimators that (nearly) minimizes $\sum_{i \in I} \text{VAR}(\tilde{w}(i))$ [24]. More precisely, Szegedy showed that the sum of per-item variances using PRI-sketches of size k is no larger than the smallest sum of variances attainable by an estimator that uses sketches with average size $k-1$.[3]

WS-sketches offer several other distinct advantages. First, they support unbiased estimators for selectivity (subpopulation fraction); Second, the estimators for selectivity and for subpopulation weight when the weight of the set is known (as in data streams), feature *negative covariances* between different items. Therefore, selectivity and weight estimators for larger subpopulations are much tighter than with the known estimator for PRI-sketches [12].

Unbiased subpopulation weight estimators exist for bottom-k sketches obtained using arbitrary rank functions [12]. These estimators are useful when we want to obtain good estimators with respect to multiple weight functions (eg, for IP flows datasets we are interested in count of distinct flows and total bandwidth).

3. MAINTAINING SKETCHES

Sketches are produced for each subset of interest in a collection of subsets over a ground set of items. The algorithms for constructing sketches are application-dependent, but on a high level,

[1]We assume to simplify the analysis that all random values are distinct.

[2]Estimators for the inverse-weight are useful for obtaining unbiased estimates for quantities where the weight appears in the denominator. These include weight ratio of two different subsets, set resemblance of two subsets, and average weight of a subset.

[3]Szegedy's proof applies only to estimators based on adjusted weight assignments. It also does not apply to estimators on the weight of subpopulations.

sketches are constructed using an incremental process, where a *current sketch* is maintained for each subset of interest, and the sketch is updated when a new information (item, or item and rank value) is presented.

We identify two operations on the current sketch, a *test* operation that checks whether incorporating the new information causes a modification of the current sketch and an *update* operation, which is a modification of the current sketch. We make the distinction between test and update because as a general rule, applications require more tests than updates, and in some applications, updates are costlier than tests.

We consider the time bounds of constructing k-mins and bottom-k sketches for two representative classes of applications. We show that when subsets are represented *explicitly* (each occurrence of an item in a subset is specified), it is much more efficient to construct bottom-k sketches. This point for uniform weights, was already noted in [4, 22]. We review it and extend the analysis for weighted items. For *implicit* representation of the subsets, via a graph, we show that the time bounds for generating the two types of sketches are comparable.

3.1 Explicit representation of subsets

Examples of applications with explicit specification are [3, 2, 22, 23]. Among these are market-basket data, Web duplicate analysis and more.

To construct a k-mins sketch for a subset, we maintain a current sketch $(m_1, \ldots, m_k)$ of the smallest rank value observed so far for each of the k rank functions (along with attributes of the items with smallest rank). Initially, $m_j = +\infty$ for $(j = 1, \ldots, k)$. When an item i is processed we compute $r^{(1)}(i), r^{(2)}(i), \ldots, r^{(k)}(i)$. We then update the sketch so that $m_j \leftarrow \min\{m_j, r^{(j)}(i)\}$. Therefore, the processing time for each occurrence of an item in a subset is $\Theta(k)$ (it is $\Theta(k)$ time for both the test and update operations).

To construct a bottom-k sketch, we use a current sketch that contains the k smallest rank values observed so far $m_1 < m_2 \cdots < m_k$ as a sorted list. When an item i is processed, we compute $r(i)$, which is compared to m_k (test operation). If $r(i) < m_k$, the rank value m_k (and corresponding item) is deleted from the list and $r(i)$ is inserted (update operation). A test operation takes $O(1)$ time and an update takes $O(\log k)$ time.

Therefore, the time bound for generating a sketch for a subset of size s is $O(sk)$ for a k-mins sketch and $O(s \log k)$ for a bottom-k sketch. We next show that for uniform weights the expected number of update operations while constructing a bottom-k sketch of a set of size s is $O(k \log s)$. This implies a better bound of $O(s + k \log s \log k)$ on the expected running time to generate a bottom-k sketch.

Lemma 3.1. *If items have uniform weights then the expected number of updates to a bottom-k sketch of a set of size s is $\leq k \ln s$.*

PROOF. A presented item triggers an update of the current sketch if and only if it has one of the bottom-k ranks among items presented so far. If j items were presented so far, the probability of that happening is $\min\{1, k/j\}$. Summing over all positions in the presentation order we obtain that the expected number of updates is at most $\sum_{j=1}^{s} k/j \approx k \ln s$. □

For weighted items we consider two cases. First is the case where items are presented in an order determined by a random permutation.

Lemma 3.2. *If items are presented in random order then the expected number of updates to a bottom-k sketch of a set of size s is $\leq k \ln s$.*

PROOF. Fix the rank assignment. The probability that the jth item in the presentation order has one of the k^{th} smallest ranks of the first j items is $\min\{1, k/j\}$. Continue as in the proof of Lemma 3.1. □

From Lemma 3.2 it follows that if items are weighted and are presented in random order, the bottom-k sketch is constructed in $O(s + k \log k \log s)$ expected time.

To bound the number of updates when items are presented in an arbitrary order we need the rank assignment to define a "close" to random permutation of the items if weights are, say, within a factor of two from each other. This will hold if the rank functions satisfy the following property.

Definition 3.3. *A family of rank functions is c-moderate if for any $w > 0$, and $0 < w' \leq 2w$, there is probability at least $\frac{1}{c}$ such that an item drawn according to $\mathbf{f_{w'}}$ has a larger rank than an item drawn according to $\mathbf{f_w}$.*

If the family of rank functions is c-moderate for some constant c and the weights of all items are within a factor of two from each other then the probability that a rank of a particular item, say i, is among the k-smallest ranks is at most $c\frac{k}{j}$, where j is the number of items.[4] One can check that exponential ranks are 3-moderate and priority ranks are 4-moderate.

Lemma 3.4. *If items are weighted and presented in arbitrary (worst-case) order, and the family of rank functions is c-moderate for some constant c, then the expected number of updates of the bottom-k sketch of a set of size s is $O(k \log(\max_i w(i)/ \min_i w(i)) \log s)$.*

PROOF. Consider a partition of the items into

$$\lceil \log(\max_i w(i)/ \min_i w(i)) \rceil$$

groups according to the weight, so that items of weight

$$[2^i \min_i w(i), 2^{i+1} \min_i w(i)]$$

are in the same group. We bound the number of updates within one group. From the fact that the rank assignment is c-moderate it follows that the probability of the jth presented item in a group to be within the bottom-k items presented so far from its group is at most ck/j, and hence, the expected number of updates within a group is at most $ck \ln s$. The statement of the lemma follows by summing over all groups. □

From Lemma 3.4 it follows that if weighted items are presented in arbitrary order, and the set of rank functions is c-moderate for some constant c, then we build the bottom-k sketch in $O(s + k \log(\max_i w(i)/ \min_i w(i)) \log s \log k)$ expected time.

3.2 Graph representation of subsets

In some applications, items and locations are embedded in a graph or a metric space and subsets correspond to all items in a certain neighborhood or the reachability set of a node [9, 13, 15]. The computation of the sketches is performed concurrently for all subsets, with items and ranks being propagated in a controlled way such that an item is tested for a subset only if it is "fairly likely" to occur in the sketch of the subset and the number of test operations is much smaller than with an explicit representation.

[4]To see that, replace item i by $\lfloor c \rfloor$ duplicates, consider a random permutation of the new set of items and the probability that one of the duplicates is among the bottom-k. This probability is smaller than $c\frac{k}{j}$ and larger than the probability that item i is among the bottom k.

We review the computation of sketches for reachability sets of nodes in a graph [9]. In this application each node is an item. Each node computes the sketch of its reachability set. Rank values (and associated information) are propagating using a graph traversal method such as breadth-first or depth-first search. When a rank value does not result in an update at a node, the propagation of the rank value is halted at that node. Therefore, the number of test operations is at most (m/n) times the number of update operations, where m is the number of edges and n the number of nodes.

For k-mins sketches, each item and a rank value associated with it are propagated separately (therefore, k truncated traversals are performed for each item). If, within each rank assignment, items are propagated in increasing rank order, then the combined number of updates for all subsets is n. Therefore, the total number of updates, for all k rank assignments and subsets is $O(kn)$ and the number of tests (and total time) is $O(km)$ [9].

Bottom-k sketches are computed by propagating each item and its associated rank using a truncated graph traversal (note that in contrast to k-mins sketches, one traversal is performed for each item). The current sketch at a node is updated when an item arrives and its rank value is smaller than the kth smallest current rank at the node. The traversal is halted at nodes where the item did not result in an update of the current sketch. When items are presented in increasing rank order, then items can only be appended to bottom-k sketches and it is never necessary to remove an item. Therefore, the total number of updates is $O(kn)$ and the total number of tests (and total time) is $O(km)$. These bounds are the same as the bounds obtained for k-mins sketches.

Arbitrary order. When items are not presented ordered by their ranks [13], the number of update operations increases. Similarly to Lemma 3.1 and Lemma 3.4 we prove that

Lemma 3.5. *Suppose we maintain the minimum rank in a subset of size s. Then*

- *if items have uniform weights and presented in a fixed but arbitrary order or if items are weighted and presented in a random order, the expected number of updates to the minimum rank is $\leq \ln s$.*
- *if items are weighted and presented in a fixed but arbitrary order and the family of rank functions is c-moderate, the expected number of updates is $O(\log(\max_i w(i)/\min_i w(i)) \log s)$.*

It follows that the total number of updates when computing k-mins sketches of all reachability sets is $O(kn \log n)$ for uniform weights and weighted items presented in random order and $O(kn \log(\max_i w(i)/\min_i w(i)) \log n)$ for weighted items presented in arbitrary order. We perform a test or update in $O(1)$ time and the number of tests is at most m/n times the number of updates. Therefore, the total time is m/n times the number of updates.

The number of updates for bottom-k sketches is given in Lemmas 3.1,3.2, and 3.4. Each update takes $O(\log k)$ time, and a test takes $O(1)$ time. The number of tests is m/n times the number of updates. Therefore, the total time is $O(\log k + m/n)$ times the number of updates given in each of these lemmas.

4. ALL-DISTANCES SKETCHES

An all-distances sketch is an encoding of plain sketches of all neighborhoods of a certain location q. For a given distance d, the sketch for the d-neighborhood of the location can be retrieved from the all-distances sketch.

We review k-mins all-distances sketches and introduce bottom-k all-distances sketches. We consider the size of the all-distances sketches, its construction time, and the time it takes to retrieve the sketch of a particular distance. We consider incremental construction, where *current all-distances sketches* are maintained and updated upon the arrival of new information (item, distance, rank). The operations we consider are *test* that determines if the current sketch needs to be modified when new information arrives, *update* of the current sketch, and *a distance query* issued to the final sketch. The distance query retrieves from the all-distances sketch the plain sketch for the neighborhood of the location q specified by the query distance.

We show that the expected size of the representation of the all-distances bottom-k sketch matches that of the k-mins sketch. When subsets are represented explicitly, the computation time of the all-distances bottom-k sketches is about factor of k faster than that of the all-distances k-mins sketches. When subsets are represented via a graph, the construction times are comparable.

All-distances k-mins sketches: We review all-distances k-mins sketches. Consider a single rank assignment. An MV/D list of a location q (Minimum Value/Distance List) encodes the minimum rank in any neighborhood (query distance) of q in a compact way. It is a list of triples where each triple contains an item e, its rank, and its distance from q. An item e is in the MV/D list of q if there is no item with smaller rank closer to q. The MV/D list is sorted in increasing distance and decreasing rank order. For a query distance d, the smallest rank of an item in the MV/D list of q of distance at most d from q is the item of smallest rank in the subset of items in the d-neighborhood of q. The expected size of the list depends on the rank function and on the weight distribution of the items.

Lemma 4.1. *The size of an MV/D list of n weighted items from a location q is bounded as follows:*

1. *When weights are uniform, the expected size is $O(\log n)$ [9].*
2. *If weights are arbitrary but items are assigned to locations at random then the expected size over assignments of items to locations, and over rank assignments is $O(\log n)$.*
3. *If items have arbitrary weights and placed in arbitrary locations and ranks are assigned using a c-moderate family of rank functions for some constant c, then the expected size is $O(\log(\max_i w(i)/\min_i w(i)) \log n)$.*

PROOF. Fix the rank assignment. Order the locations in increasing distance from q. The assignment of items to location defines a random permutation of the ranks. Therefore, the probability that the rank value in location j is smaller than the rank values in all closer locations (and therefore the item occurs on the MV/D list) is $1/j$. By summing over all positions, we obtain that the expected size of the MV/D list is $\sum_{j=1}^{n} 1/j \approx \ln n$. □

If the relation of the weights and the locations of items is arbitrary, the expected size of the MV/D lists depends on the location of items: If item weights are decreasing with distance then the expected size of the MV/D list is smaller and if item weights are increasing with distances, then the expected size is larger (can be linear in the worst case). The worst-case size of the MV/D list, however, can be bounded by the weight distribution of the items. The proof of the following lemma is similar to that of Lemma 3.4.

Lemma 4.2. *If items have arbitrary weights and placed in arbitrary locations and ranks are assigned using a c-moderate family of rank functions for some constant c, the expected size of the MV/D list is $O(\log(\max_i w(i)/\min_i w(i)) \log n)$.*

PROOF. Let $w_1 = \min_i w(i)$. Consider a partition of the items so that all items with weight in $[w_1 2^i, w_1 2^{i+1})$ are in group i, for $i = 0, \lfloor \log_2(\max_i w(i)/\min_i w(i)) \rfloor$. By the property of c-moderate rank functions, the expected number of items from each group that appear on the MV/D list is logarithmic in its size. Therefore, the total expected number of items on the MV/D list is bounded by

$$2 \ln n (1 + \ln(\max_i w(i)/\min_i w(i))) \ .$$

□

The MV/D list can be constructed incrementally: When presented with a new item, its rank, and distance, the list is updated only if the new item has smaller rank than all items on the list that have the same or smaller distance. If items are presented in order of increasing rank, (or increasing (distance,rank) in lexicographic order), then items are never removed from the list during updates [9]. Other orders of presenting items were analyzed in [13]. We summarize and extend these results in the following lemma.

Lemma 4.3. *Assume that we construct an MV/D list of a location q, and there are n weighted items. Then,*

1. *When items are presented in random order and there are uniform weights, the expected number of updates is $O(\log^2 n)$ [13].*
2. *If items are assigned to locations at random, the expected number of updates to the MV/D list, over assignments of items to locations, rank assignments, and presentation order of items is $O(\log^2 n)$.*
3. *If ranks are assigned using a c-moderate family of rank functions for some constant c, then the expected number of updates to the MV/D list, over rank assignments, and presentation order of items is $O(\log(\max_i w(i)/\min_i w(i)) \log^2 n)$.*

All-distances bottom-k sketches: An all-distances bottom-k sketch encodes the bottom-k items in a neighborhood defined by any query distance from a location q. The all-distances bottom-k sketch is a data structure that generalizes a single MV/D list. An item i, its rank value $r(i)$, and distance $d(i)$ are represented in the sketch if and only if the item has one of the bottom-k ranks in the $d(i)$-neighborhood of the location.

It is convenient to think of the all-distances bottom-k sketch as a list of lists arranged by increasing distance. For each distance d where the set of bottom-k items within distance d changes, we record the list of bottom-k items within this distance. This list is valid until the next distance for which there is a change.

The list of lists representation, however, is not storage efficient, since all but one item are repeated in two consecutive lists. This sketch can be more compactly represented if we only record the changes to the list. In Section 5 we discuss compact representations for an all-distances bottom-k sketch that require storage proportional to the number of distances where the bottom-k set changes.

We bound the number of distances for which the bottom-k list changes. These bounds imply that the storage for an all-distances bottom-k sketch is comparable to the storage for k MV/D lists in an all-distances k-mins sketch.

Lemma 4.4. *Consider an all-distances bottom-k sketch for n items of a location q. We bound the expected number of distances from q where the set of bottom-k items changes.*

1. *For uniform weights, the expected number of distances is $O(k \log n)$.*
2. *For a set of items with arbitrary weights that are randomly assigned to locations the expected number of distances (over assignments of items to locations, and over rank assignments) is $O(k \log n)$.*
3. *If items have arbitrary weights and placed in arbitrary locations and ranks are assigned using a c-moderate family of rank functions for some constant c, the expected number of distances is $O(k \log(\max_i w(i)/\min_i w(i)) \log n)$.*

PROOF. Order the items by increasing distance from q. Let $d(j)$ be the distance of the jth item in this order from q. The jth item is in the bottom-k set of items within distance $d(j)$ from q if it is one of the k-smallest items among the j closest items to q. Since weights are uniform, the ranks define a random permutation of the items which is independent of the their distances to q. So the jth item is among the smallest k with probability $\min\{k/j, 1\}$. Summing over all items we obtain that the expected number of items which are among the kth smallest items within their distance from q is at most

$$\sum_j \frac{k}{j} \approx k \ln n$$

As in Lemma 3.1, and 3.4 for weighted items we can show the following.

Lemma 4.5.
1. *For a set of items with arbitrary weights and a set of locations, the expected number of distances from a location q where the set of bottom-k items changes, over assignments of items to locations, and over rank assignments is $O(k \log n)$.*
2. *If items have arbitrary weights and placed in arbitrary locations and ranks are assigned using a c-moderate family of rank functions for some constant c, the expected number of distances from a location q where the set of bottom-k items changes is $O(k \log(\max_i w(i)/\min_i w(i)) \log n)$.*

If items are presented in order of increasing distances from q we can obtain a bottom-k list for the current distance, from the bottom-k list of the previous distance by doing an insertion and a deletion. Similarly, if items arrive sorted by rank value, then the number of updates to the bottom-k sketch is proportional to the size (number of breakpoint distances) of the sketch. We can also bound the number of updates performed if items arrive in a random order.

Lemma 4.6. *Consider the expected number of updates that is performed in an incremental construction of an all-distances bottom-k sketch of a location q when items are presented in a random order (the order is a random permutation)*

1. *When item weights are uniform, the expected number of updates is $O(k \log^2 n)$.*
2. *When items have arbitrary weights, the expected number of updates over assignments of weights to locations, over rank assignments, and arrival order, is $O(k \log^2 n)$.*
3. *When items have arbitrary weights, and the family of rank functions is c-moderate, the expectation over rank assignments and arrival orders of the number of updates is $O(k \log(\max_i w(i)/\min_i w(i)) \log^2 n)$.*

PROOF. Consider uniform weights (Part 1). An item would result in an update if at the time it is presented, it has one of the k smallest ranks amongst items already presented that are at least as close to q. Consider the jth closest item to q. It has probability $1/j$ of having the ith rank among all items that are at least as close to the location. We now calculate the probability that the item results in an update given that it has the ith rank. Consider the $i-1$ items that have smaller ranks and are at least as close. The probability that at most $k-1$ of them are presented before our item is that of being in one of the first k positions in a random permutation of i items, which is $\min\{k/i, 1\}$. We obtain that the expected number of updates for the jth closest item is $\sum_{i=1}^{j} \min\{k/i, 1\}/j \leq (1/j)\sum_{i=1}^{j} k/i \approx (k/j)\ln j$. summing over all n items, we obtain that the expected number of updates is

$$\leq \sum_{j=1}^{n} (k/j)\ln j \leq k \ln^2 n \, .$$

The proof of Part 2 and Part 3 follows by an argument as for Lemma 3.1, and Lemma 3.4.

As in the case of a single sketch in Section 3 the number of test operations depends on the representation of the subsets. If this representation is explicit then since k-mins sketch consists of k independent MV/D lists the number of tests required for a k-mins sketch is by a factor of k larger than for a bottom-k sketch. In a graph representation, the number of tests is at most (m/n) times the number of updates for both kinds of sketches. In Section 5 we discuss representations of sketches that allow efficient implementations of test and update operations.

5. REPRESENTATIONS OF SKETCHES

We consider possible representations for k-mins sketches and bottom-k sketches. We are interested in bounding the size of the data structure that encodes the sketch, and the time required to incrementally construct the sketch when items are presented in sorted or other orders. For all-distances sketches we also consider the time it takes to find the sketch for a particular query distance.

Representation of an MV/D list: An efficient data structure for an MV/D list construction and querying was not explicitly discussed in earlier works. If items arrive sorted, by increasing rank value or increasing distance, we represent an MV/D list sorted by increasing distances (and decreasing ranks), as a binary search tree. With this representation we can support distance queries in expected $O(\log M)$ time, where M is the expected size of the list.

If items do not arrive in a sorted order, we represent the current MV/D list as a dynamic binary search tree. Test operations then require expected $O(\log M)$ time. An update is performed in $O(\log M)$ expected amortized time: Each item requires an insertion to the tree if it has the smallest rank within its distance from the query location, and possibly a series of deletions of items which are further away from the query location and of larger rank. Since each item can be deleted at most once, we can charge each deletion to the respective insertion.

The all-distances k-mins sketches consists of k independent MV/D lists, one for each rank assignment. Therefore, for any query distance, we can obtain the min-rank sketch over the items that lie within that distance in $O(k \log M)$ time, by searching independently in each of the k lists. The query time can be improved to $O(k + \log M)$ using fractional cascading [7]. Using fractional cascading, we perform a binary search only on one list and use links between items to find the position in the next list is $O(1)$ time.

Another approach to obtain a $O(k + \log M)$ bound per query is to use an interval tree or a segment tree (See e.g. [16]) to represent the kM intervals defined by consecutive points on the same list. We can then do stabbing queries to find the k intervals of a query distance, which correspond to the min-rank in that neighborhood in each of the k rank functions.

Constructing and querying the bottom-k sketch: A natural representation for a single bottom-k sketch is a list of the items sorted by increasing ranks represented as a search tree, as mentioned in Section 3. However for all-distances bottom-k sketch one needs to be more careful so that the size of the representation would be proportional to the number of distances where the list changes as mentioned in Section 4. We suggest possible efficient representations for an all-distances bottom-k sketch.

Ordered insertion of items: When items are presented in an order related to their distances or ranks, we can use the following data structures.

If items are presented in order of increasing distances from q we can obtain a bottom-k list for the current distance, from the bottom-k list of the previous distance by doing an insertion and a deletion. If we use a persistent list [17] to represented each bottom-k list, then we can update a bottom-k list to obtain the next one in $O(k)$ time while consuming only $O(1)$ space. We can reduce the update time to $O(\log k)$ by using persistent search trees instead of persistent lists, the space required per operation is still $O(1)$.

We can also construct the bottom-k all-distances sketch if items are presented in order of increasing ranks so that it takes space proportional to the number of updates. We construct the first list after the k items with smallest ranks are presented. This list is associated with the distance of the item among these k which is furthest from the query location q. When the next item arrives, say item j, if item j is closer to q than any of the already seen items, we construct a new bottom k list L. Assume that the previous list L' which we constructed was associated with distance $d > d(j)$. We construct L from L' by deleting from L' the item at distance d from q and adding item j instead. The distance associated with L is the distance of the furthest item in L from q. Using persistent lists or persistent search trees to represent the bottom-k lists we construct all lists in space which is proportional to the number of updates. The update time is $O(k)$ with persistent lists and $O(\log k)$ with persistent trees (we keep the items in each list sorted by increasing distances from q).

Insertion of items in arbitrary order: To support arbitrary insertion order, we can think of the all-distances bottom-k sketch as a set of intervals on a line. Each item corresponds to an interval over the range of distances in which it is a bottom-k item. Let D be the current set of intervals. A query is a point stabbing query, the bottom-k list consists of the set of intervals in D intersecting the query point.

When a new item z arrives at distance d we should figure out if the sketch should be updated. Let $I_1 = [d_1, d_2)$ be the interval spanning distance d with the largest rank. We should update the sketch if the rank of z is smaller than the rank of the item corresponding to I_1. We update the sketch as follows. We replace I_1 with $I_1' = [d_1, d)$. Then we find the interval $I_2 = [d_2, d_3)$ with largest rank at distance d_2. If the rank of I_2 is larger than the rank of z we delete I_2, and we continue in the same way finding for $i > 2$ the interval I_i of largest rank at distance d_i, and deleting I_i if the rank of the corresponding item is larger than the rank of z. Let d_j be the right endpoint of the last interval which we deleted. We insert the interval $[d, d_j)$ corresponding to item z. Since each interval is inserted and deleted once the total number of insertions and deletions of intervals is proportional to the number of intervals. An interval I may split many time. However, each split of I is associated with a newly inserted interval immediately following I. Since each inserted interval may cause at most one split the total number of splits is also proportional to the total number of intervals.

To support these interval operations, we can maintain the intervals either in a dynamic interval tree or in a dynamic segment tree [8]. Let M denote the number of intervals in the tree. A dynamic interval tree takes $O(M)$ space, and using it we can report the k intervals stabbed at a particular distance in $O(\log(M)\log(k) + k)$ time. We can update an interval tree in $O(\log(M)\log(k))$ amortized time. A dynamic segment tree requires $O(M \log M)$ space and supports queries in $O(\log(M) + k)$ time and updates in $O(\log(M)\log(k))$ amortized time.

By a standard modification to an interval tree in which we store at every secondary node the item of maximum rank in its subtree we can find the interval of maximum rank stabbed by a query distance in $O(\log(M)\log(k))$ time. Similarly, by maintaining at each node of a segment tree the maximum rank interval that it contains we can find the maximum rank interval stabbed by a query distance in $O(\log(M))$ time. This allows us to test if the bottom-k sketch changes when a new item arrives in polylogarithmic time. (This is in contrast with $O(k\log(n))$ time for k independent MV/D lists that form a k-mins all distances sketch.)

6. MIMICKED SAMPLING WITH REPLACEMENT

We present a randomized procedure that uses a WS-sketch (weighted sampling without replacement until k items are obtained) to emulate weighted sampling with replacement. Using this process, we can derive a size-k WSR-sketch from a size-k WS-sketch. By mimicking we mean that the probability to obtain a particular sketch by first obtaining a WS-sketch and then applying the procedure is the same as when directly obtaining a WSR-sketch.

The process is described as generating a sequence of items (and rank values). The process is randomized and therefore every WS-sketch b corresponds to a distribution $M(b)$ over such sequences. If we stop the process after k samples, we obtain a WSR-sketch. We can use a different stopping rule and continue until the $(k+1)$ *distinct* item is sampled. We refer to a weighted sample with replacement with this stopping rule as a WSRD-sketch. The WSRD-sketch contains the same set of items as the WS-sketch but also has a count for each item that corresponds to the number of times the item is sampled until the process is stopped.

Mimicking allows us to apply an estimator ν designed for WSR-sketches or WSRD-sketches to WS-sketches. A WS-sketch estimator can be obtained by drawing a mimicked sketch $s \in M(b)$ using this process and returning $\nu(s)$. This estimator is equivalent to using the estimator ν on WSR or WSRD-sketches.

The estimator $\nu'(b) = E(\nu(s)|s \in M(b))$ has lower variance (a consequence of Lemma 2.3). It can be approximated [5] by taking average of $\nu(s)$ over multiple draws of $s \in M(b)$.

Lower variance estimator (another consequence of Lemma 2.3) is obtained by considering the subspace $L(b)$ of WS-sketches with the same subset of items as b and if $w(J)$ is not provided and the same rank value r_{k+1}. $L(b)$ is an equivalence relation that defines a partition of the sample space. The estimator $\nu''(b) = E(\nu'(b')|b' \in L(b))$ can be approximated by averaging $\nu(b')$ over multiple draws of $b' \in L(b)$.

We first provide a mimicking process when the total weight $w(I)$ of the ground set is known. Let $i_1, \ldots, i_k$ be the items in the WS-sketch b, ordered by increasing ranks. The first item in the mimicked sample is i_1. We then select i_1 with probability $w(i_1)/w(I)$ and i_2 otherwise, and repeat this until we have k samples or until i_2 is selected. In phase j, after outputting at least one sample of each of $i_1, \ldots, i_j$, we select i_ℓ with probability $w(i_\ell)/w(I)$ (for $1 \le \ell \le j$) and i_{j+1} otherwise. Each phase can be simulated efficiently using the geometric distribution to determine the number of samples until the "next" item from b is sampled and the multinomial distribution to determine the number of times each item is sampled.

[5] This "approximation" preserves unbiasedness.

We now provide a mimicking procedure when $w(I)$ is not known. The procedure is applied to an ordered sketch where all items have rank values.

We use properties of the exponential distribution and the ranks of the items in the WS-sketch. We first establish few lemmas about the distribution of the differences between the ranks of the items in a WS-sketch. The first lemma follows from the memoryless nature of the exponential distribution.

Lemma 6.1. *Consider a subspace of rank assignments where the order of the items according to rank values is fixed, say $i_1, \ldots, i_n$, and the rank values of the first j items are fixed. Let $r(i_{j+1})$ be the random variable that is the $(j+1)$st smallest rank. The conditional distribution of $r(i_{j+1}) - r(i_j)$ is exponential with parameter $\sum_{h=j+1}^{n} w(i_h)$.*

PROOF. Since rank values of different items are independent, the probability density for the event: *items $i_1, \ldots, i_j$ have the bottom-j ranks with the values $r(i_1) < \cdots < r(i_j)$ and items $i_{j+1}, \ldots, i_n$ having the next $n-j$ smallest ranks in that order* is the product $p_1 p_2$ where

$$\begin{aligned} p_1 &= w(i_1)\exp(-r(i_1)w(i_1))w(i_2)\exp(-r(i_2)w(i_2)) \\ &\quad \cdots w(i_j)\exp(-r(i_j)w(i_j)) \end{aligned}$$

(probability density that the items $i_1, \ldots, i_j$ have the rank values $r(i_1), \ldots, r(i_j)$) and

$$\begin{aligned} p_2 &= \int_{r(i_j)}^{\infty} w(i_{j+1})\exp(-x_{j+1}w(i_{j+1})) \\ &\quad \cdot \int_{x_{j+1}}^{\infty} w(i_{j+2})\exp(-x_{j+2}w(i_{j+2})) \\ &\quad \cdots \int_{x_{n-1}}^{\infty} w(i_n)\exp(-x_n w(i_n))dx_n \cdots dx_{j+2}dx_{j+1} . \end{aligned}$$

is the probability density that items $i_{j+1}, \ldots, i_n$ have rank values in that order and all larger than $r(i_j)$. Performing the integration, we obtain that

$$p_2 = p_3 \exp(-r(i_j)\sum_{h=j+1}^{n} w(i_h)) ,$$

where

$$p_3 = \frac{w(i_{j+1})}{\sum_{h=j+1}^{n} w(i_h)} \frac{w(i_{j+2})}{\sum_{h=j+2}^{n} w(i_h)} \cdots \frac{w(i_{n-1})}{w(i_{n-1}) + w(i_n)} .$$

(p_3 is the probability that the rank values of items $i_{j+1}, \ldots, i_n$ are in that order and $\exp(-r(i_j)(\sum_{h=j+1}^{n} w(i_h)))$ is the probability that the minimum rank among $i_{j+1}, \ldots, i_n$ is at least $r(i_j)$.) Therefore, the probability density is

$$p_1 p_2 = p_1 p_3 \exp\left(-r(i_j)\sum_{h=j+1}^{n} w(i_h)\right) . \qquad (1)$$

We next calculate the probability density for the following event: *items $i_1, \ldots, i_n$ have increasing ranks, the bottom-j ranks are equal to $r(i_1) < \ldots < r(i_j)$, and the $(j+1)$st rank has value $r(i_j) + d$.* It follows from independence of the rank values that the probability density is

$$\begin{aligned}
& p_1 w(i_{j+1}) \exp\left(-(r(i_j)+d)w(i_{j+1})\right) \\
& \int_{r(i_j)+d}^{\infty} w(i_{j+2}) \exp\left(-x_{j+2} w(i_{j+2})\right) \\
& \cdots \int_{x_{n-1}}^{\infty} w(i_n) \exp(-x_n w(i_n)) dx_n \cdots dx_{j+2} \\
= \;& p_1 w(i_{j+1}) \exp\left(-(r(i_j)+d)w(i_{j+1})\right) \\
& \exp\left(-(r(i_j)+d)\sum_{h=j+2}^{n} w(i_h)\right) \\
& \frac{w(i_{j+2})}{\sum_{h=j+2}^{n} w(i_h)} \cdots \frac{w(i_{n-1})}{w(i_{n-1})+w(i_n)} \\
= \;& p_1 \left(\sum_{h=j+1}^{n} w(i_h)\right) \exp\left(-(r(i_j)+d)\sum_{h=j+1}^{n} w(i_h)\right) \\
& \frac{w(i_{j+1})}{\sum_{h=j+1}^{n} w(i_h)} \frac{w(i_{j+2})}{\sum_{h=j+2}^{n} w(i_h)} \cdots \frac{w(i_{n-1})}{w(i_{n-1})+w(i_n)} \\
= \;& p_1 p_3 \exp\left(-r(i_j)\sum_{h=j+1}^{n} w(i_h)\right) \\
& \left(\sum_{h=j+1}^{n} w(i_h)\right) \exp\left(-d\sum_{h=j+1}^{n} w(i_h)\right) \quad . \qquad (2)
\end{aligned}$$

The density function of the conditional probability distribution in the statement of the lemma equals to the ratio of Eq. (2) and Eq. (1). This ratio is

$$\left(\sum_{h=j+1}^{n} w(i_h)\right) \exp\left(-d\sum_{h=j+1}^{n} w(i_h)\right) ,$$

which is the probability density of the exponential distribution with parameter $\sum_{h=j+1}^{n} w(i_h)$ at d. $\square$

In the following corollary we relax the conditioning of Lemma 6.1 to what we need.

Corollary 6.2. *Consider a probability subspace of rank assignments such that the permutation of the first k items as determined by the rank order is fixed to be $i_1, i_2, i_3, \ldots, i_k$. Let $r(i_1) < r(i_2) < \cdots < r(i_k) < r(i_{k+1})$ be the random variables that are the smallest $k+1$ ranks. The rank differences $r(i_1), r(i_2)-r(i_1), \ldots, r(i_{k+1})-r(i_k)$ are independent random variables, where $r(i_j)-r(i_{j-1})$ ($j=1,\ldots,k+1$) is exponentially distributed with parameter $w(J) - \sum_{\ell=1}^{j-1} w(i_\ell)$. (we formally define $r(i_0) \equiv 0$.)*

The WS-sketch provides the k-prefix of the random permutation defined by the ranks. By Corollary 6.2 the rank differences $r(i_j) - r(i_{j-1})$ are k independent samples where the jth sample is from an exponential distribution with parameter $w(J) - \sum_{h=1}^{j-1} w(i_h)$, for $i = 1, \ldots, k$. The following lemma (which states a basic property of the exponential distribution) allows us to transform an exponentially distributed random variable drawn with parameter $A - a$, where $A \geq 0$ is not known but a ($A \geq a \geq 0$) is known, to an exponentially distributed random variable with parameter A.

Lemma 6.3. *Let a be an exponentially distributed random variable with parameter A. Let r be an independent exponentially distributed random variable with parameter $B-A$ (for some $B > A$). Then the random variable $\min\{a, r\}$ is exponentially distributed with parameter B.*

Let $i_1, \ldots, i_k$ be the items in the sketch in increasing rank order.

- The first entry of the mimicked sketch contains the item i_1 with rank value $r(i_1)$.
- Suppose items $i_1, \ldots, i_j$ ($j \geq 1$) are drawn (at least once). The next entry is obtained as follows:
  - We draw exponentially distributed values $r'_{j,\ell}$ from distributions with parameters $w(i_\ell)$ (for $1 \leq \ell < j$).
  - If $\min_{1 \leq \ell < j} r'_{j,i} > r(i_j) - r(i_{j-1})$ we use the item i_j with rank value $r(i_j) - r(i_{j-1})$.
  - Otherwise, let $m = \arg\min_{1 \leq \ell < j} r'_{j,\ell}$ and use the item i_m and the rank value $r'_{j,m}$.

When implementing this process, we can use a geometric random variable (with parameter equal to the probability that an exponential random variable with parameter $\sum_{h=1}^{j} w(i_h)$ is at most $r(i_j) - r(i_{j-1})$) to determine the number of draws until another distinct items is sampled and multinomial random variables to determine the number of items each item is sampled.

The following lemma summarizes the basic property of this randomized process. Its correctness follows from Corollary 6.2 and Lemma 6.3.

Lemma 6.4. *The mimicked sketches have the following property: For a subset J, the following two processes yield the same distribution over sketches.*

1. *Draw items using weighted sampling with replacement and assign independent rank values from an exponential distribution with parameter $w(J)$ until: k samples are obtained (for mimicked WSR-sketches) or the $(k+1)$st distinct item is sampled (for mimicked WSRD-sketches).*
2. *Generate a single rank assignment, derive a WS-sketch b of size k and draw a mimicked sketch from $M(b)$ as above.*

6.1 Weight estimation

Consider a set $J \subset I$ and a WS-sketch b of size k. Let $i_1, i_2, \ldots, i_k$ be the items in b and $r(i_1) < \cdots < r(i_k)$ be their rank values. We consider the problem of estimating $w(J)$ from the sketch b. If the cardinality of the set J is at most $|J| < k$ then the WS-sketch contains all the elements of the subset (and we can determine this) and we can compute $w(J) = \sum w(i_j)$. When $|J| \geq k$, we apply the estimator to the mimicked sketch.

The ranks in a WSR-sketch are k independent exponentially distributed random variables with parameter $w(J)$. This property was used in [9] to obtain sketch-based estimators for $w(J)$: If $v_1, \ldots, v_k$ are independent and exponentially distributed with parameter $w(J)$ then $\frac{k-1}{\sum_{h=1}^{k} v_h}$ is an unbiased estimator of $w(J)$ with standard deviation equal to $w(J)/\sqrt{k-2}$ and average (absolute value of the) relative error approximately $\sqrt{2/(\pi(k-2))}$ [9]. $\frac{\sum_{h=1}^{k} v_h}{k}$ is an unbiased estimator of $1/w(J)$ with standard deviation $1/(\sqrt{k}w(J))$.

The WSRD mimicking process produces $\ell \geq k$ independent random variables $v_1, \ldots, v_\ell$. The number $\ell \geq k$ is a random variable that is independent of the values. It is not hard to see that if $k > 1$, $\frac{\ell-1}{\sum_{h=1}^{\ell} v_h}$ is an unbiased estimator of $w(J)$ and if $k > 2$, its standard deviation is at most $w(J)/\sqrt{k-2}$. Similarly, $\frac{\sum_{h=1}^{\ell} v_h}{\ell}$ is an unbiased estimator of $1/w(J)$ with standard deviations at most $1/(\sqrt{k}w(J))$.

Figure 1 shows the (absolute) relative error of different estimators, averaged over 1000 runs. Bottom-k estimators performs better than the k-mins WSR-sketches estimator. There tailored bottom-k estimators are derived in [11, 12] (exponential ranks) and [1] (priority ranks).

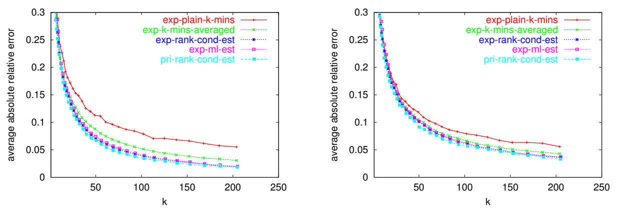

Figure 1: Average relative error of estimators on item weights for 1000 items drawn from Pareto distributions with $\alpha = 0.9$ and $\alpha = 1.3$. The estimators shown are the plain k-mins (WSR-sketches), the k-mins averaged (on WS-sketch b, averaged over $M(b)$ but not over $L(b)$), and some tailored bottom-k estimators.

6.2 Selectivity

The *Selectivity* of a subpopulation $J' \subset J$ is $w(J')/w(J)$, that is, the weighted fraction of the J' items included in J.

An important application of selectivity is computing the *resemblance* of two subsets, defined as $w(A \cap B)/w(A \cup B)$ from their sketches[5]. (Resemblance generalizes the binary Jaccard coefficient). The resemblance is the selectivity of items in $A \cap B$ that are included in $A \cup B$: The sketch of $A \cup B$ can be obtained from the sketches of A and B (for both k-mins and bottom-k sketches). For each item in the sketch of $A \cup B$ we can determine if it is a member of A and of B (it is a member in A if it is in the sketch of A and symmetrically for B) and hence if it is a member of $A \cap B$. Mimicked sketches do not support subset relations, and therefore, when applying WSR or WSRD estimators we have to first compute the WS-sketch of the union and obtain a mimicked sketch of the union.

When items have uniform weights, an unbiased estimate of selectivity (and resemblance) can be obtained as follows. Using k-mins sketches, the estimator is the fraction of entries in the sketch that contain members of the subpopulation (for resemblance the fraction of identical entries in the sketches of A and B) [9, 5]. Using bottom-k sketches, the estimator is the fraction of items in the sketch that are members of the subpopulation (fraction of items in the sketch of $A \cup B$ that are in $A \cap B$) [4, 22]. The k-mins selectivity estimator carries over for weighted data [13]: The fraction of entries that are members of the subpopulation is an unbiased selectivity estimator. We provide a simple example that demonstrates that this estimator is biased for bottom-k sketches when items are weighted. Consider a set of four items (i_1, i_2, i_3, i_4) with weights $(4, 1, 1, 1)$ and estimating the selectivity of $\{i_1\}$. (For resemblance, consider the subsets $\{i_1, i_2, i_3\}$ and $\{i_1, i_4\}$ – the union contains all four items and the intersection contains only $\{i_1\}$.) The selectivity is $4/7$. Consider $k = 2$. The probability that i_1 appears (first or second) in the sketch is $4/7 + (3/7) * (4/6) = 6/7$ in that case, the respective fraction is $4/5$ (since the other item in the sketch has weight 1). Otherwise, i_1 does not appear in the sketch and the fraction is zero. Therefore, the expectation of the fraction is $(6/7)(4/5) > 4/7$. If we use the fraction of entries instead of fraction of weights, we obtain $3/7 < 4/7$.

When the weight $w(J)$ is not provided, (for resemblance, we do not have the weight $w(A \cup B)$ even if we have $w(A)$ and $w(B)$), unbiased estimators for selectivity from WS-sketches do not follow from existing unbiased estimators for subpopulation size [1, 12]. Fortunately, unbiased selectivity (and resemblance) estimators can be obtained using mimicked sketches. An unbiased selectivity estimator for WSR-sketches (and also for WSRD-sketches [12]) is the fraction of samples where the item is a member of the subpopulation. A WS-estimator is obtained by taking the expectation of this estimator over mimicked sketches. This is the only method we are aware of to obtain unbiased estimators of selectivity and resemblance from WS-sketches that have variance that is at most that obtained through a WSR-sketch.

7. REFERENCES

[1] N. Alon, N. Duffield, M. Thorup, and C. Lund. Estimating arbitrary subset sums with few probes. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems*, pages 317–325, 2005.

[2] K. Bharat and A. Z. Broder. Mirror, mirror on the web: A study of host pairs with replicated content. In *Proceedings of the 8th International World Wide Web Conference (WWW)*, pages 501–512, 1999.

[3] A. Broder. Filtering near-duplicate documents. In *FUN*, 1998.

[4] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29. ACM, 1997.

[5] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, volume 1848 of *LLNCS*, pages 1–10. Springer, 2000.

[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[7] B. Chazelle and L. Guibas. Fractional cascading: I. a data structuring technique. *Algorithmica*, 1(2):133–162, 1986.

[8] Y.-J. Chiang and R. Tamassia. Dynamic algorithms in computational geometry. *Proceedings of the IEEE*, 80(9):1412–1434, 1992.

[9] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.*, 55:441–453, 1997.

[10] E. Cohen and H. Kaplan. Efficient estimation algorithms for neighborhood variance and other moments. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2004.

[11] E. Cohen and H. Kaplan. Bottom-k sketches: Better and more efficient estimation of aggregates. In *Proceedings of the ACM SIGMETRICS'07 Conference*, 2007. poster.

[12] E. Cohen and H. Kaplan. Sketches and estimators for subpopulation weight queries. Manuscript, 2007.

[13] E. Cohen and H. Kaplan. Spatially-decaying aggregation over a network: model and algorithms. *J. Comput. System Sci.*, 73:265–288, 2007.

[14] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *Proc. of the 2003 ACM Symp. on Principles of Database Systems (PODS 2003)*. ACM, 2003.

[15] E. Cohen, Y.-M. Wang, and G. Suri. When piecewise determinism is almost true. In *Proc. Pacific Rim International Symposium on Fault-Tolerant Systems*, pages 66–71, December 1995.

[16] M.T. de Berg, O. Schwarzkopf, M.J. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[17] J. R. Driscoll, N. Sarnak, D. Sleator, and R. Tarjan. Making data structures persistent. *J. of Computer and System Science*, 38:86–124, 1989.

[18] N. Duffield, M. Thorup, and C. Lund. Flow sampling under hard resource constraints. In *Proceedings the ACM IFIP Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, pages 85–96, 2004.

[19] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. System Sci.*, 31:182–209, 1985.

[20] H. Kaplan and M. Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar voronoi diagrams, and approximate range counting. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 484–493, New York, NY, USA, 2006. ACM Press.

[21] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings of the ACM PODC'06 Conference*, 2006.

[22] R. Motwani, E. Cohen, M. Datar, S. Fujiware, A. Gronis, P. Indyk, J. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13:64–78, 2001.

[23] N. T. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Proceedings of the ACM SIGCOMM'00 Conference*. ACM, 2000.

[24] M. Szegedy. The DLT priority sampling is essentially optimal. In *Proc. 38th Annual ACM Symposium on Theory of Computing*. ACM, 2006.