MODELING RANDOMNESS IN NETWORK TRAFFIC

- LAVANYA JOSE, INDEPENDENT WORK FALL'11

ADVISED BY PROF. MOSES CHARIKAR

ABSTRACT. Sketches are randomized data structures that allow one to record properties of huge streams of items using little memory and processing time. Some sketches require truly random hash functions in theory while they do well enough with simpler universal hash functions in practice. Mitzenmacher and Vadhan [3] explained that this was because the analysis assumed worst case data, when in practice data was somewhat random. They explained how to quantify this idea by viewing the data as a block source. Then universal hash functions 'extract' the randomness in the blocks i.e. the 'conditional entropy per block' to produce almost truly random hash values according to the sketch's need.

I used their techniques to find the minimum 'conditional entropy per block' of the data one must assume for distinct counting sketches (the *Log Log counter* [2].) Then, I analyzed real network traces to see if this assumption was reasonable. I also study two new 'average' measures of randomness in the traffic, which are easier to estimate in practice.

1. THELOG LOG COUNTER AND A GAP BETWEEN THEORY AND PRACTICE

The *Log Log counter* [2] is a randomized data structure (or 'sketch') that can estimate " the number of different words in the whole of Shakespeare's works" in one pass in just a few bytes.

It uses a hash function to map an incoming item (or word) to one of m 'counters', each of which maintains a rough sketch of the number of distinct items that hashed to it. Each individual counter keeps track of the most significant bit of the hash value [1]he suffix following the bucket index among all the items that mapped to it. The *Log Log counter's* simple scheme and good guarantees make it amenable to network measurement applications such as counting distinct flows or ports at line rate in switches.

Date: 2012-01-10.

[1]t

The scheme guarantees that its estimate approaches the actual distinct count asymptotically. However for this it requires that the hash function be truly random. This means each item is mapped uniformly to one of the counters and completely independently of the remaining items. Similarly the hash values over which the maximum most significant bit are computed should also be uniform on the range and completely independent of each other, so that the maximum is a good estimate of the distinct items mapped to the counter.

However a truly random hash function requires an exponential number of bits to describe. All practical implementations of Log Log counters use hash functions with much weaker independence (such as 2 or 4 universal hash functions.)

Fortunately, it turns out that in practice, the performance of the *Log Log counter* even using just universal functions matches the performance of truly random hash functions.

2. Worst case data versus Real World Data

Mitzenmacher and Vadhan identified the reason behind the discrepancy. Most analyses of sketches assume worst case data, whereas real world data has enough randomness to even make up for the limited independence of universal hash functions. For the *Log Log counter* Durand and Flajolet "appeal to a hashing function in order to randomize data and bring them to a form that resembles random (uniform, independent) binary data." They ignore the possibility of extracting randomness from the data so that their analysis would work for any kind of data. Practical implementations that use universal hash functions on the other hand must extract the randomness in the data to hash close to the uniform distribution, that sketches like the *Log Log counter* require.

3. Modeling Randomness in the Traffic- Using Block Sources

3.1. **From Data with high 'Collision Entropy Per Block' to Hash Values with 'Low Collision Probability'.**

Mitzenmacher and Vadhan's main contribution lies in quantifying the notion of randomness in the data using the block sources and model and showing how much "collision probability per block" is needed for a random universal hash function to hash the block to as close

to the uniform distribution as needed. They use 'Renyi entropy' (also called 'collision entropy') which is $-log(\text{collision probability})$ as their notion of randomness. They note that it is 'a less stringent measure than min-entropy' traditionally used to analyze block sources. They prove the following main theorem-

3.2. **Theorem 1.** Let $H : [N] \to [M]$ be a random hash function from a 2-universal family H. For every block source $(X1, ..., XT)$ with collision probability $1/K$ per block and every $\epsilon > 0$, the random variable $Y = (H(X1), ..., H(XT))$ is ϵ-close to a block source Z with collision probability $1/M + T/(\delta K)$ per block. In particular, if $K \geq MT^2/\delta$, then Z has collision probability at most $(1 + 2MT^2/(K\delta))M^T < 3/M^T$.

They show that if each item of a block source X has collision probability at most $1/K$ (i.e. at least $-lg(K)$ collision entropy or unpredictability) conditioned on the previous items, then a random hash function H from its domain to $[M]$can extract this minimum entropy to yield hash values $Y = H(X_1)...$ where each Y_i has c.p. at least $1/\delta K + 1/M$ with high probability $1 - \delta$ over H.

They then use Y and the hash function to construct a block source Z, that is $T\delta$ close to Y in statistical distance and because of Y has c.p. not more than $1/K + 1/M$ per block conditioned on the previous blocks. From this they conclude that if $K > 2MT/\delta$, Z's global collision probability too is within a constant factor of uniform (i.e. $1/M^T$.)

3.3. **Finding the minimum 'collision prob. per block' required for Applications.** They also show the effect of $T\delta$ statistical distance between Y and Z on the expected difference in estimates or functions f computed from them. They derive a corresponding result for the effect 'collision probability distance' between Z and U on the $E[f(Z)] - E[f(U]$. This lends their results to be easily applied to different applications where performance metrics are typically functions of just the hash values. For e.g., we can use their result to find the expected difference in the distinct count estimate of a Log counter if it used uniform hash values (i.e., with a truly random hash function) versus if it used universal hash values derived from block sources like X above. In fact before I

understood how general their result was, I spent a considerable amount of time delving into the details of why *Log Log counters* work to identify where exactly the truly random assumption was being used. While this was enlightening, I could also have simply used their general result to get an upper bound on $E[f(Y)] - E[f(U]$ without knowing much about the worst case analysis. This would work for most applications where the performance metric is a function f of the hash values (such as Linear Probing, Bloom Filters [3] and Linear Counter [5].)

3.4. **Finding the minimum 'collision prob. per block' required for Log Log counters.** We can use the above theorem, the definition of statistical difference and Lemma 3 given below (from [3]) to bound the expected difference between distinct count estimates using universal hash values $E[f(Y)]$ versus a uniform distribution $E[f(U)]$. From this, we can get the maximum collision probability $1/K$ needed for the two estimates to be comparable. This technique is similar to the one used for the Linear Probing application in [3], where they find the maximum collision probability so that the insertion times as functions of Y and of U are comparable. In fact since the range of the hash function and the maximum value of their functions f (insertion time for Linear Probing, estimated distinct count for *Log Log*) are $O(T)$ for both, the dependence of K on T also turns out to be similar for both.

3.4.1. *Definition 2: Statistical difference.* For random variables Y and Z taking values in $[N]$, their statistical difference is defined as $\max_{S\subset[N]} \mathrm{P}\{Y \in S\} - \mathrm{P}\{Z \in S\}$. They are ϵ close if this difference is $\leq \epsilon$.

3.4.2. *Lemma 3: Bounding difference between $E[f(X)]$ and μ in terms of X's collision probability 1/K.* If X takes value in $[M]$ and f is a function from $[M]$ to $[N]$ for every such function $|\ [E[f(Z)] - E[f(U)]]\ | \leq \sigma\sqrt{M/K}$ where U is the uniform distribution on $[M]$ and σ^2 is its standard variance of Eu. [3]

3.4.3. *Applying the main theorem to get distance between hash values Y and U.* Let $E[f(Y)]$ be the expected estimated form the hashed values Y. According to the theorem Y is ϵ close to a block source Z, where Z has collision probability $< 1/M^T(1+2MT^2/(\epsilon K))$ (if $K > MT^2/(\epsilon K)$)

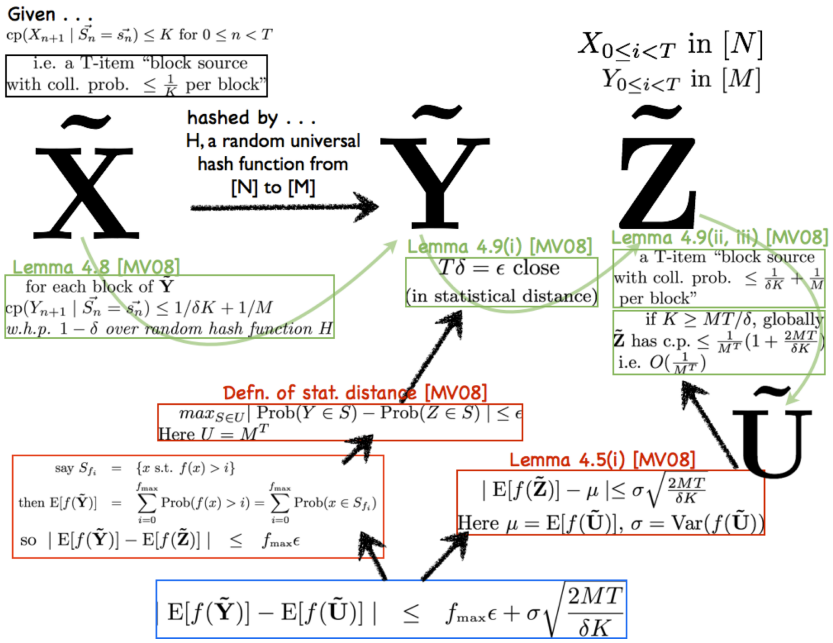

FIGURE 1. Sketch of Proof for bounding difference in Expected Estimates from Y and U

3.4.4. *Using Definition 1 of Stat. Difference to bound* $E[f(Y)] - E[f(Z)]$.

$$
\begin{aligned}
| \ [E[f(Y)] - E[f(Z)]] \ | \ &\leq \ \sum_{i=1}^{f_{max}} \mathrm{P}\{f(Y) > i\} - \mathrm{P}\{f(Z) > i\} \\
&\text{Let } S_{f_i} \text{ be the set of values x for which } f(x) > i \\
&\leq \ \sum_{i=1}^{f_{max}} \mathrm{P}\{Y \in S_{f_i}\} - \mathrm{P}\{Z \in S_{f_i}\} \\
&\text{Since Y is } \epsilon \text{ apart from Z, by Definition 1} \\
&\leq \ f_{max}\epsilon \\
&\text{The estimate can take maximum value } f_{max} = T_{max} \\
&= \ T_{max}\epsilon
\end{aligned}
$$

3.4.5. *Using Lemma 3 to bound* $E[f(Z)] - E[f(U)]$. The main theorem tells us that Z has $c.p. < 1/M^T(1 + 2MT^2/(\epsilon K))$, then we can use Lemma 3 to bound $E[f(Z)] - E[f(U)]$ Here σ^2 is the variance of the estimate for a truly random hash function.

$$\mid [E[f(Z)] - E[f(U)]] \mid \leq \sigma\sqrt{2MT^2/(\epsilon K)}$$

3.4.6. *Range of the hash function for the Log Log.* The hash function for the *Log Log counter* indexes into one of m buckets and has enough bits left over so that the 'number of trailing zeroes plus one' estimates $lg(T/m)$ which could possibly be as large as $lg(T_{max})$, the maximum distinct items we could get. Durand and Flajolet suggest allocating three extra bits to minimize the collisions for their truly random hash values in practice (though collisions don't matter as much for us. If the sequence of distinct data items have enough entropy, then our hash values can be made close enough to uniform even if some are identical which is quite likely for universal hash functions when there are many items.) Then our range M needs $lg(T_{max}) + 3 + lg(m)$ bits i.e. $M = 8mT_{max}$. Then

$$\begin{aligned} \mid [E[f(Z)] - E[f(U)]] \mid &\leq \sigma\sqrt{2MT^2/(\epsilon K)} \\ &= \sigma\sqrt{16mT^3/(\epsilon K)} \end{aligned}$$

3.4.7. *Using the above bounds on $E[f(Y)] - E[f(Z)]$ and $E[f(Z)] - E[f(U)]$ to bound difference in expected estimate using hash values Y and expected estimated using uniform hash values from a truly random hash function.*

$$\begin{aligned} \mid [E[f(Y)] - E[f(U)]] \mid &\leq \mid [E[f(Y)] - E[f(Z)]] \mid + \mid [E[f(Z)] - E[f(U)]] \mid \\ &\leq T_{max}\epsilon + \sigma\sqrt{16mT^3/(\epsilon K)} \end{aligned}$$

ϵ must be $O(T_{max})$ for the first term to be small, this means $K = O(16m{T_{max}}^4\sigma^2)$

The variance for a truly random hash function σ^2 asymptotically approaches $1/m$ [2], we can conclude $K = O({T_{max}}^4)$.

3.4.8. *Implications.* This bound is quite high for applications such as counting distinct ports. Note that the $1/K$ is the collision probability, so K can only take as many values as are in the range of X i.e. $K \leq R$. Then if $K = O(T_{max}^4)$, T_{max} must be less than $R^{1/4}$. For ports, which have a range of 2^{16}, the turns out to be just $2^4 = 16$. However for applications such as counting distinct flows (the four tuple $srcip, dstip, srcport, dotport$), the range is quite large and the number of distinct items T_{max} so the items have enough entropy can be reasonably large. In addition, we could also use the tighter bound on $E[f(Z)] - E[f(U)]$ proved by [1] to bring down the required K to $O(T_{max}^3)$.

3.5. **Verifying using traces: measuring the maximum 'collision prob. per block'.** The values of K are definitely feasible for distinct counting applications such as counting distinct ports on a link (though it limits T severely) or distinct flows. Intuitively it captures several ideas such as the little long range dependencies that might be there because of which instead of we don't simply state the condition probability is $1/N$. This leads us to wonder whether we can verify this from real traces, and justify once and for all that this is the way to model randomness in real traffic.

This is almost impossible.

One of the key problems lies in the very definition of a block source. To verify that a T item data sequence can be modeled as a block source with collision probability k per block, we need to check that for every $i < T$ and for every possible value x of X the collision probability of X_i conditioned on $x < i$ is at most $1/K$. When estimating from a sample, this requires that not only do we see all possible values of X that we expect in our ideal distribution but also that we have enough examples if each possible $x < i$ so we can estimate its conditional probability accurately. An estimate such as the *maximum* conditioned collision probability is especially sensitive to this. For e.g., if we have only one example for a particular value $x < i$, then we would estimate the

maximum collision probability as 1! Ideally, we would like an estimate that not only scales or becomes more accurate with the sample size, but is also easy to estimate.

4. Modeling Randomness in the Data- using $cp(X_i \mid X_{<i})$ weighted by $\mathrm{P}\{X_{<i} = x_{<i}\}^2$

This leads us to look at other statistics that can convey the randomness in the data in terms of the conditional collision probability of the ith item. But this time we account for the possibility that we may not have enough examples of some x_i, for which which our estimates of $cp(X_i \mid X_{<i} = x_{<i})$may be far from their real values. We would like their contribution to our estimate be minimum. We note that the more examples a value of x_i the more accurate their collision probability estimate would be. So it seems like a good idea to weight the contribution of each i to collision probability estimate $\hat{cp}(X_{n+1} \mid S_n)$by a function of $\mathrm{P}\{S_n =\}$. Thus we use the following measure-

$$(\textstyle\sum_i \mathrm{P}\{S_n = i\}^2 \hat{cp}(X_{n+1} \mid S_n = i))/\sum_i \mathrm{P}\{S_n = i\}^2$$

Our estimate is robust to insufficient examples, gives a rough idea of the collision probability per block and is probably can also be shown to guarantee some maximum collision probability for the hash values using a universal hash function along the lines of the block source model.

4.1. Verifying over traces: measuring the weighted (by $\mathrm{P}\{S_n = i\}^2$) average 'collision prob. per block'.

There is another big advantage to using this new measure weighted by $\mathrm{P}\{S_n = i\}^2$ rather than $\mathrm{P}\{S_n = i\}$ (which would correspond to the expected value of $\hat{cp}(X_{n+1} \mid S_n = i)$ over X_T) The former statistic is equal to the the ratio of the F_2 norm of S_{n+1} to the F_2 norm of S_n. Here F_2 norm of S_n is $\sum_{i \in Supp(S_n)} {n_i}^2$ where n_i is the frequency of item i. These *aggregate* functions of the $n+1$ and $n$item tuples in the traffic can each be computed accurately in little space. Sketches such as those described in [4] can estimate F2 to within ϵ using just O($1/\epsilon^2$) space .

$$\begin{aligned}
\frac{\sum_i \mathrm{P}\{S_n = i\}^2 \hat{cp}(X_{n+1} \mid S_n = i)}{\sum_i \mathrm{P}\{S_n = i\}^2} &= \frac{\sum_i (n_i/N)^2 \sum_j \mathrm{P}\{X_{n+1} = j \mid S_n = i\}^2}{\sum_i (n_i/N)^2} \\
&= \frac{\sum_i {n_i}^2 \sum_j \mathrm{P}\{X_{n+1} = j \mid S_n = i\}^2}{\sum_i n_i^2} \\
&= \frac{\sum_i {n_i}^2 \sum_j (n_{ij}/n_i)^2}{\sum_i n_i^2} \\
&= \frac{\sum_{ij} n_{ij}^2}{\sum_i n_i^2} \\
&= F2(S_n, X_{n+1})/F2(S_n) \\
&= F2(S_{n+1})/F2(S_n) \\
&\quad \text{labeled in fig 2 as } Cn + 1/Cn
\end{aligned}$$

4.2. **Experiment.**

4.3. **Method.** To evaluate the trends in the weighted collision probability estimate, I used a ten minute packet trace (around 280 million packets in all) collected at a backbone link of a Tier-1 ISP in San Jose, CA, at 12 pm on Sept. 17, 2009. I considered the 'block source' values x one would get from the trace for a distinct counting application counting $T = 10$ distinct items. I listed the first ten distinct port numbers for each possible packet my measurement could start from. I split this list of around 280 million values for X into intervals of around four million each. I then computed the the F_2 sketch for S_i for $1 \leq i \leq 10$ each interval separately.

The plot shows the value of $Cn + 1/Cn$ for n on the Y axis, and the length of the interval in units of (4 million packets) on the X axis. One trend that stands out is that the entropy estimates for C2/C1 and C3/C2 are high while the entropy seems to be close to zero for C4/C3 onwards. This seems surprising considering that the estimate for C4/C3 is the sum of estimates $\hat{cp}(X_4 \mid S_3 = s_3)$ weighted by the (square of) probability of seeing s_3 , so that s_3s which have inaccurate estimates like 0 because they were only seen once should contribute less to the overall estimate.

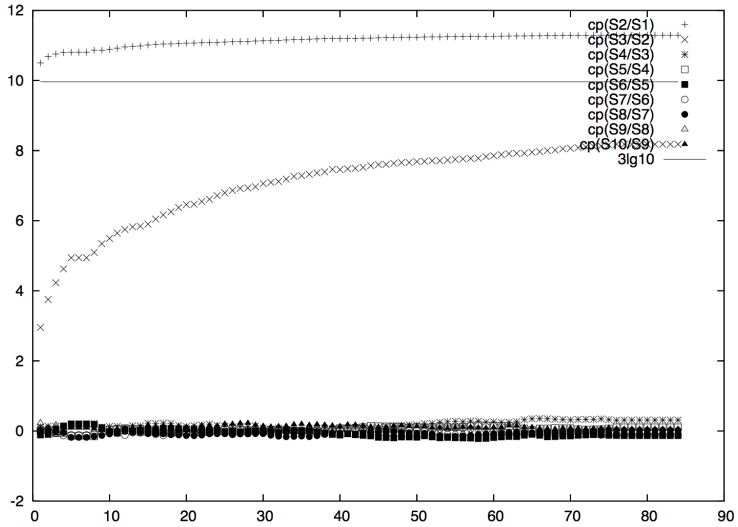

FIGURE 2. 10 second trace for Distinct Counting, $Cn + 1/Cn$ for $1 <= n < 10$ on the Y axis, sample interval length in units of 4 million packets on the X axis

However the problem is that the trace is too short (280 million packets) for any of the C3s to appear more than once. (e.g., compare $C3 = F_2(S_3)$ over the whole trace of 28 million instances is 387 million while $C2 = F_2(S_2)$ is 112 billion, this indicates many duplicates values occur among the 28 million S_2 instances but there are hardly any duplicates among the S_3)

Thus each value of S_3 seen contributes zero (collision) entropy to the overall estimate, because of which our estimate is close to 0. However as the trace size gets large enough, a few values of S3 appear more than once, and the (collision) entropy estimate for $S3/S4$ will increase. In conclusion, it appears that we will need a lot of data (though not an impossibly large amount) even for reasonable *average* measures of conditional collision probability.

5. MODELING RANDOMNESS IN THE DATA- USING EXPECTED VALUE OF $cp(X_i \mid X_{<i})$ OVER X_T

A third metric for quantifying randomness in the data is the expected value of $cp(X_i \mid X_{<i})$ over X_T. This is also feasible to estimate from a sample trace though perhaps not efficiently like the previous metric. We can also use this measure in a way similar to that for the original metric

$\max cp(X_i \mid X_{<i})$ to extract the entropy via universal hash functions and get close to uniform hash values in theory.

5.1. **Verifying using traces.** We could try to use sketches to estimate this as well, since $E\left[\hat{cp}(X_{n+1} \mid S_n)\right]$ is the F2 sketch of each appearance of $S_n, X_{n+1} = ij$ weighted by $1/\sqrt{(}n_i)$.

$$\begin{aligned}\sum_i \mathrm{P}\left\{X_{n+1} = j \mid S_n = i\right\} &= \sum_i n_i \, {n_{ij}}^2 / {n_i}^2 \\ &= \sum_i {n_{ij}}^2 / n_i \\ &= \sum_{ij} (n_{ij}/\sqrt{(}n_i))^2\end{aligned}$$

However we need accurate frequencies even for the (possibly many) items with low frequencies, and sketches wouldn't be able to capture all the information. Thus we would need to resort to sorting algorithms instead to compute individual n_is. Another possibility is that we use short sketches to estimate the n_i so that they are more accurate.

5.2. **How to get close to uniform hash values.** We can also relate the expected value of $cp(X_i \mid X_{<i})$ over X_T to the distance between the hash values and the uniform distributions. (Along the lines of Lemma 4.9 in [3]) The difference is that here rather than fixing an upper bound on the conditional collision probability over all i, we let an 'average measure of the conditional collision probability' be bounded instead. We can use the Markov inequality then to find the probability that for any i, $cp(X_i \mid X_{<i})$ is more than the $1/K$ allowed for close to uniform. An average measure like this will allow us to account for situations where $cp(X_2 \mid X_{<2})$ may be high (short range dependencies are high), but $cp(X_i \mid X_{<i})$ for longer i is may be lower. However it looks like the Markov inequality doesn't give us a very strong result since q_i, the probability that any x_i's estimate c.p. is too high equals $E\left[cp(X_i \mid X_{<i})\right] K$. So the expected value of coll. prob. for

any i from our estimate $E\left[cp(X_i \mid X_{<i})\right]$ would have to be very low with the average less than O($1/K^2$).

(1) From our experiments we can get an estimate of the $E\left[cp(X_i \mid X_{<i})\right]$ for the random variable $cp(X_i \mid X_{<i})$ over X_T.

This will let us calculate an upper bound q_i for $\mathrm{P}\left\{cp(X_i \mid X_{<i}) > 1/K\right\}$

$$\mathrm{P}\left\{cp(X_i \mid X_{<i}) > 1/K\right\} \quad \leq \quad E\left[cp(X_i \mid X_{<i})\right]K = q_i$$

(2) Now $\mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\} = cp(X_i \mid X_{<i}) = C$ is a random variable over X_T taking values in $(0, 1)$ and we know $\mathrm{P}\left\{cp(X_i \mid X_{<i}) > 1/K\right\} < q_i$

So

$$\begin{aligned}
E\left[C\right] &= E\left[\mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\}\right] \\
&= \int_{c=0}^{1} c\,\mathrm{P}\left\{C \in dc\right\} \\
&= \int_{c=0}^{1/K} c\,\mathrm{P}\left\{C \in dc\right\} + \int_{c=1/K}^{1} c\,\mathrm{P}\left\{C \in dc\right\} \\
&\leq 1/K \int_{c=0}^{1/K} \mathrm{P}\left\{C \in dc\right\} + 1 \int_{c=1/K}^{1} \mathrm{P}\left\{C \in dc\right\} \\
&\leq 1/K\,\mathrm{P}\left\{C \leq 1/K\right\} + \mathrm{P}\left\{C > 1/K\right\} \\
&\leq 1/K + q_i
\end{aligned}$$

(3) Using this we can find $E\left[cp(H(X_i) \mid X_{<i})\right]$ over H and X_T

$$\begin{aligned}
E\left[cp(H(X_i) \mid X_{<i})\right] &= \mathrm{P}\left\{H(X_i) = H(X'_i) \mid X_{<i}, X_i = X'_i\right\} \mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\} \\
&\quad + \mathrm{P}\left\{H(X_i) = H(X'_i) \mid X_{<i}, X_i \neq X'_i\right\} \mathrm{P}\left\{X_i \neq X'_i \mid X_{<i}\right\} \\
&= \mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\} + 1/M(1 - \mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\}) \\
&= \mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\}(1 - 1/M) + 1/M \\
&= E\left[\mathrm{P}\left\{X_i = X'_i \mid X_{<i}\right\}\right](1 - 1/M) + 1/M \\
&\leq (1/K + q_i)(1 - 1/M) + 1/M \\
&\leq 1/K + q_i + 1/M
\end{aligned}$$

(4) Consider the random variable $Z = cp(H(X_i) \mid X_{<i}) - 1/M$

$$\begin{aligned}
\mathrm{P}\left\{Z > 1/\delta K\right\} &< E\left[Z\right](\delta K) < \delta(1 + Kq_i) \\
\text{So } \mathrm{P}\left\{cp(H(X_i) \mid X_{<i}) > 1/\delta K + 1/M\right\} &< \delta(1 + Kq_i) \text{ over } H \text{ and } X_T
\end{aligned}$$

(5) Since this holds for all i, given a random universal hash function H and a random T-item vector X, we can show that for $\delta \in (0, 1)$

$\mathrm{P}\left\{\exists i : cp(H(X_i) = H(X'_i) \mid X_{<i}) > 1/(\delta)K + 1/M\right\} < \sum_i (q_i K + 1)\delta$

(6) Define the random variable $Y = H(X_1), H(X_2), ..., H(X_T)$ where $X_1, X_2, X_3, ..., X_T$ is a sequence seen in our sample and let $\epsilon = \delta T$

Then we can use a proof along the lines of Lemma 4.9's proof (second part, "Define a random variable Z . . .") and use the fact above to construct a block source Z that is $(1 + \sum_i q_i/T\,K)\epsilon$ close Y and has collision probability $1/M + T/(\epsilon K)$ per block. If $K >= MT^2/\epsilon$, then Z has collision probability at most $((1 + 2MT^2)/(\epsilon K))/M^T$.

(7) Now for the distance between Y and Z to be $O(\epsilon)$, we need $\sum_i (q_i)/T = O(1/K)$.

6. Conclusion

In conclusion, we started with the block source model for explaining how randomness in data makes up for limited independence. We applied its results to the distinct counter sketch to find the minimum collision probability per block so a random universal function performs as well a a truly random hash function. The model is intuitive, captures trends in real data and quantifies the relation between its metric and the distance of the hash values to uniform clearly. Moreover it yields general techniques for comparing the performance of applications using truly random hash values or universal hash values from a random enough block source.

However the metric was not practically verifiable form real traces. It required that we have examples of every possible vector. It was very sensitive to inaccuracies in the conditional probability estimate (which arose because of insufficient examples.) Thus we proposed two other notions of 'average' randomness in data also based on the block source model, which took into account the inaccuracies due to insufficient examples. The first one (probability square weighted collision probability) can be estimated scalably, while the second estimate (expected conditional collision probability) can be related to distance of the hash values from uniform. We also estimate probability square weighted collision probability from real a 10 minute trace for short distinct item sequences. We found that though it is easy to estimate and gives a good idea of the conditional entropy, we still need a lot of data to get reasonable estimates even though we're using an 'average' measure .

7. Discussion

7.1. **The right sample size.** It is not clear whether we should wait for estimates of a sketch to converge, or whether once they're high enough to match our metrics, we can stop and assume that a distribution computed over that size is always enough.

If our sample is too small, this conditional collision probability estimate may be too high, because many of the s_n seen may not have enough examples to get a good low enough estimate for the conditional collision prob. of X_{n+1} following it. As result, their contribution to the expected value may be too high.

As we get a bigger sample, the conditional collision probability (estimate) should tend to decrease we can expect to see newer values for S_n and also expect the estimated collision probabilities for old s_n to decrease rapidly as more different instances of X_{n+1} are seen following it.

On the other extreme, if we have an infinitely large sample, the expected value (especially for large n) should tend to 1/N. This is because the conditional probability of $X_n + 1$ for each s_n seen should be roughly uniform, as there doesn't seem to be much correlation between say (for n=20) the first 20 items of a distinct vector and the 21st item). However not only is it impractical to get an infinitely large sample, it makes more sense for X_n to take values from among those that have actually been at a link during some time.

7.2. **The right metric for modeling randomness in the data.** In this paper we have discussed three ways of quantifying the randomness in the data based on the block source model.

They differ in terms of how easy it is to extract entropy from data to get close to uniform hash values, how practically verifiable they are and how scalably they can be estimated.

Since they are all based on the notion of collision probability of the itth item conditioned on the previous $i - 1$ items, they all capture trends in real data such as- there is could be dependencies over a long range of items as well as over short sequences.

7.3. **Open Questions.** There are several questions that my project raised and I would like to understand better. I don't yet know how the probability- square weighted average of $cp(X_i \mid X_{<i})$ estimate relates to the uniformity of hash values, though I have been able to estimate it easily and accurately from real traces. On the other hand I don't know any way to compute the expected value of $cp(X_i \mid X_{<i})$ scalably enough, though I can relate it to the uniformity of the hash values.

Ideally we want our model to be easily justified by both theory and from traces, and also be intuitive.

REFERENCES

[1] K.-M. Chung and S. P. Vadhan. Tight bounds for hashing block sources. *CoRR*, abs/0806.1948, 2008.

[2] M. Durand and P. Flajolet. Loglog counting of large cardinalities. In *ESA*, 2003.

[3] M. Mitzenmacher and S. Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. In *SODA*, 2008.

[4] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 615–624, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[5] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Syst.*, 15:208–229, June 1990.