HLL-based TV panel audience extrapolation compatible with online audience measurement from Logs

Shen-fu Tsai, Evgeny Skvortsov, Jim Koehler

July 2021

1 Introduction

Advanced online audience measurement has come a long way. Koehler, Skvortsov, and Vos (2013) [3] (KSV) presents a method for measuring reach and frequency of online ad campaigns by audience attributes for one device (or cookie) type by combining ad server logs, publisher provided user data (PPD), census data, and a representative panel to produce corrected cookie and impression counts by these audience attributes. Koehler, Skvortsov, Ma, and Liu (2016) [2] (KSML) extends the method to today's world of multiple device types such as desktop, smartphone, and tablet with a formulation for converting multiple cookie counts to people counts. Skvortsov and Koehler (2019) [6] (KS) further presents a technology that implements the measuring methodologies [3] and [2] in large scale production systems efficiently, where reach and demographic correction models are converted into assignments of Virtual People for each events in the logs. Each Virtual Person has demographic attributes (age and gender) assigned to them. Total reach of an audience (ad campaign, web site, online video etc) can be estimated as a simple count of unique Virtual People assigned to the corresponding set of events. The demographic composition of an audience is then estimated as the demographic composition of the set of Virtual People.

HyperLogLog [4, 1], or HLL sketch, is a data structure that stores a *sketch* of incoming objects to facilitate cardinality estimation. Let N and M be the maximum number of distinct objects and number of registers of an HLL sketch, respectively. The sketch takes $O(M \log \log N)$ space, much smaller than the naive approach that takes $O(N \log N)$ space to store all

distinct incoming objects. HyperReal [7], or HR sketch, is an extension to HyperLogLog that enables cardinality slicing by object attribute.

This paper proposes an extension to [6] for cross-media audiences, where TV audiences are measured via extrapolation from a panel or partial set-top-box data. In essence, a set of TV Virtual People are exclusively associated with, or represented by, a TV panelist q via an HR sketch s, which is an extension to an HLL sketch. We extrapolate q's TV activity to this set of Virtual People, thus s serves as an input to the system and can be deduplicated with the digital part of the audience via a simple sketch merge.

The main contribution of this paper is an efficient method that takes as input Q panelists and $\mathcal{P}$, the union of Virtual People they represent, and assigns an HR sketch to each panelist. The efficiency-accuracy trade-off is controlled by a depth parameter D, and to help decide D in practical systems we provide an upper bound on the accuracy loss due to a finite depth D. For example, to have an error of no more than 1% we can set D to 9. The size of the deep sketch is roughly proportional to the depth.

The rest of the paper is organized as follows. In Section 2 we briefly introduce HyperLogLog and HyperReal to clarify notations and facilitate the discussion. In Section 3 we give formal definition of the problem to be solved, including the goals that we want to achieve. In Section 4 we describe a naive algorithm that is correct but usually too slow in practice. In Section 5 we propose our solution: **deep sketch sampling**. In particular we prove an upper bound on the error of cardinality estimation, thus showing its effectiveness and giving guidance on how to select the parameter, i.e. depth D.

2 HyperLogLog and HyperReal

2.1 HyperLogLog

An HLL sketch[4, 1] is initialized with a fixed number of registers. Each incoming object o is hashed to one of the registers, and then further mapped to a uniform integer hash $H(o)$ represented with m bits where m is fixed. A register either stores a maximum number of leading zeros of all incoming uniform integers, called **register value**, or could be empty if no object is distributed to it. The more leading zeros the smaller $H(o)$ is. Obviously, duplicate incoming objects will go to the same register and then mapped

to the same number of leading zeros. Cardinality estimation is based on these numbers of leading zeros across registers. If N is the maximum number of distinct objects, then we need $m = \Theta(\log(N))$ for hash $H(\cdot)$ to be conflict-free. Since the number of leading zeros is at most m and it takes $O(\log(m)) = O(\log(\log(N)))$ bits to store it, it is now clear where the name *HyperLogLog* comes from.

To simplify notation, **for the rest of the paper we assume there is only one register unless stated otherwise,** and our approaches and analyses extend naturally to multiple registers. Algorithm 1 illustrates how to generate an HLL sketch for a set of objects $\mathcal{O}$. To merge two HLL sketches, we take the maximum of the sketch values as described in Algorithm 2.

Algorithm 1: $GenerateHLL(\mathcal{O})$ computes an HLL sketch for a set of objects $\mathcal{O}$

input : Set of objects $\mathcal{O}$
output: HLL sketch $GenerateHLL(\mathcal{O})$

```
r ← NULL
for o ∈ O do
    if r is NULL or r < m − ⌊log2 H(o)⌋ then
        r ← m − ⌊log2 H(o)⌋
    end
end
return r
```

2.2 HyperReal

HyperLogLog estimates cardinality, but what if we also want to know how it is sliced by certain attributes, say demographic buckets that take z values, without creating z HLL sketches? In [7] an extension to HyperLogLog was proposed to address it. That paper also presented an algorithm called HyperReal, which was aimed to give a better intuition about HyperLogLog. Here we observe that extra data stored in HyperLogLog for sampling purposes can be interpreted as amounting to a real number value, and thus the extended HyperLogLog can be intuitively interpreted as an efficient representation of HyperReal. We will also use this real value for sampling

Algorithm 2: $MergeHLL(r_1, r_2)$ computes the merge of HLL sketches r_1 and r_2

input : HLL sketches r_1 and r_2
output: HLL sketch $MergeHLL(r_1, r_2)$

```
if r_1 is NULL then
    return r_2
end
if r_2 is NULL then
    return r_1
end
return max(r_1, r_2)
```

sub-sketches efficiently.

The method leverages registers that far outnumber z to store the distribution of attributes at the same time. To better understand it, we first note that the register value of HyperLogLog described in Section 2.1 is equivalent to the number of leading zeros after the decimal point of a uniform hash $h(\cdot) \equiv \frac{H(\cdot)}{2^m} \in (0, 1)$, and the expression of leading zero count becomes $-\lfloor \log_2 h(\cdot) \rfloor$. From the perspective of HyperLogLog, each register also stores the attribute of the object whose hash has the most leading zeros, and to break a tie we compare another uniform hash g which thus has to be stored as well. Note that because $\frac{h(\cdot)}{2^{\lfloor \log_2 h(\cdot) \rfloor}}$ is uniform in $[1, 2)$, we can set $g = \frac{h(\cdot)}{2^{\lfloor \log_2 h(\cdot) \rfloor}}$, which together with the number of leading zeros represent nothing other than $h(\cdot)$, the original uniform hash. This explains the name *HyperReal*: a floating number approximating a uniform hash in $(0, 1)$ is stored per register. HyperReal brings intuition about how HyperLogLog works, and reasoning about $h(\cdot)$ is much easier than the number of leading zeros. In practice, rather than full 4-byte floating numbers, the exponent and significand of $h(\cdot)$ take one byte each so we store $h(\cdot)$ in a total of two bytes. In the example illustrated in Figure 1, we use $2^{14} = 16384$ registers, each represented by a vertical bar.

Since slicing cardinality by demographic buckets is essential in our application, for the rest of the paper we base our discussion on HyperReal rather than HyperLogLog. It is then straightforward to visualize each register as a vertical bar storing a minimum uniform hash as illustrated in Figure 1. Moreover, distributing objects across registers is straightforward.

Algorithm 3 and Algorithm 4 summarize the generation and merge of HyperReal, respectively.

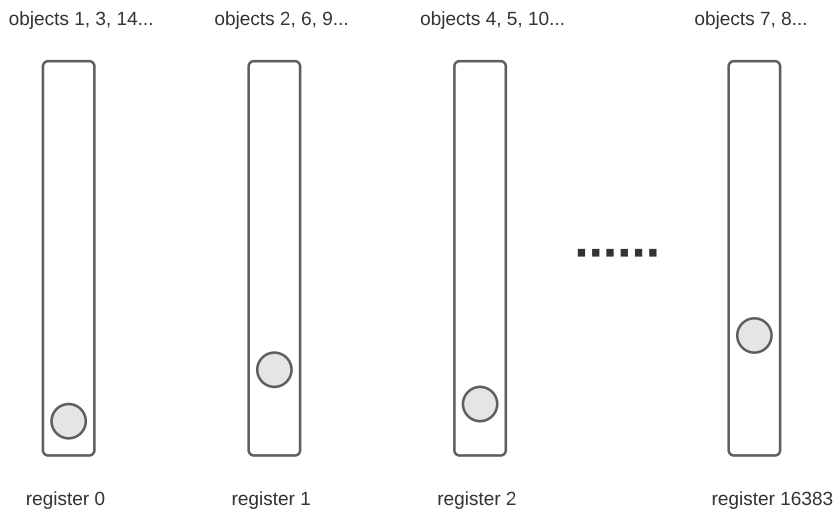

Figure 1: Distinct objects are distributed across registers to form an HR sketch, and each register stores the minimum of uniform hashes of the objects it receives. The circle in each vertical bar denotes the smallest uniform hash of the objects falling into that register.

3 Problem definition

We are given Q panelists numbered $1, 2, \ldots, Q$ with non-negative weights $w = (w_1, w_2, \ldots, w_Q)$ and a set of Virtual People $\mathcal{P}$. All panelists and Virtual People have the same attributes like geo location, age, gender, etc. For simplicity of notation, assume $w_1 + \ldots + w_Q = 1$. We want to associate panelists with Virtual People such that

- Each Virtual Person p is associated with exactly one panelist chosen by $PickPanelist(h(p), w)$ that takes uniform hash $h(p) \in (0, 1)$ of p as input. In other words, we are seeking a partitioning of $\mathcal{P}$ into Q disjoint sets. This is because we want to make each panelist represent

Algorithm 3: $GenerateHR(\mathcal{O})$ computes an HR sketch for a set of objects $\mathcal{O}$

input : Set of objects $\mathcal{O}$
output: HR sketch $GenerateHR(\mathcal{O})$

```
r ← ∞
for o ∈ O do
    r ← min(r, h(o))
end
return r
```

Algorithm 4: $MergeHR(r_1, r_2)$ computes the merge of HR sketches r_1 and r_2

input : HR sketches r_1 and r_2
output: HR sketch $MergeHR(r_1, r_2)$

```
return min(r_1, r_2)
```

a subset of Virtual People, and naturally these subsets should not overlap and should union to the whole set $\mathcal{P}$.

- Each panelist is associated with Virtual People proportional to their weight. Specifically, $Prob(PickPanelist(h(p), w) = q) \approx w_q$ when p is picked uniformly at random from $\mathcal{P}$. Although the requirement seems vague, obviously it is natural when we interpret $w_q|\mathcal{P}|$ as the rough number of Virtual People panelist q represents.

- **Stability requirement**. The association is relatively stable with respect to panelist weights, i.e. a small change to $w_1, w_2, \ldots, w_Q$ should keep $PickPanelist(h(p), w)$ the same for most $p \in \mathcal{P}$. Though we are again being a bit vague here, in practice since panelist weights could fluctuate and panelists would come and go (modeled by weights switching between zero and non-zero values) over time this is also a desired property to avoid assigning Virtual People to different panelists more often than necessary. We will further discuss how our algorithms address the requirement in Section 4.

- **Joinability requirement**. Once the first two requirements are met by certain association, we will be able to project a subset of the Q panelists to a subset $\mathcal{P}_1$ of $\mathcal{P}$. For $\mathcal{P}_1$ to properly *interact* with any other independent subset $\mathcal{P}_2$ of $\mathcal{P}$, naturally we would like the cardi-

nality estimate of $\mathcal{P}_1 \cup \mathcal{P}_2$ based on their HR sketches to be as close to the expected value $|\mathcal{P}_1| + |\mathcal{P}_2| - \frac{|\mathcal{P}_1||\mathcal{P}_2|}{|\mathcal{P}|}$ as possible, whether or not $\mathcal{P}_2$ is obtained through the association.

Finally, for practical purpose we do not store all Virtual People associated with a given panelist. Instead, we want to have an HR sketch r_q for panelist q that represent his or her associated Virtual People.

4 Naive association

Were it not for the **stability requirement** described above, we could just assign roughly $w_q|\mathcal{P}|$ to panelist q for $1 \leq q \leq Q$ and be done. We present *affinity hashing* algorithm [5] below that satisfies the stability requirement.

Algorithm 5: $PickPanelist(h(p), w)$ samples a panelist from multinomial distribution $w_1, w_2, \ldots, w_Q$

input : Panelist weights $w = (w_1, \ldots, w_Q)$ and Virtual Person p
output: Panelist $PickPanelist(h(p), w)$
1 **return** $\arg\min_{1 \leq q \leq Q} \frac{-\ln h'(h(p),q)}{w_q}$

In Algorithm 5, $h'(h(p), q)$ is a hash function that maps Virtual Person p's uniform hash $h(p)$ and panelist index q to a number uniformly distributed in $(0, 1)$, and therefore $-\ln h'(h(p), q)$ is exponentially distributed.

Conceptually, each Virtual Person p generates Q independent exponential random variables with rates $w_1, w_2, \ldots, w_Q$ respectively, and associates with the panelist corresponding to the minimum of those exponential random variables. Because with probability $\frac{w_q}{\sum_{q=1}^{Q} w_q}$ panelist q is the index of the variable that achieves the minimum [8], we have

$$Prob(PickPanelist(h(p), w) = q) = \frac{w_q}{\sum_{i=1}^{Q} w_q} = w_q.$$

Moreover, it could be seen that as weights change a bit so do the exponential random variables since the hashing function $h'(\cdot)$ is fixed, therefore the assignment would mostly stay the same.

Once each Virtual Person picks a panelist, we can generate an HR sketch for every panelist based on the set of Virtual People that picks them, as

shown in Algorithm 6. It should be clear that the resulting HR sketches meet all requirements listed in Section 3.

Algorithm 6: $NaiveAssociate(w, \mathcal{P})$ produces a sketch r_q for each panelist q

input : Panelist weights $w = (w_1, \ldots, w_Q)$, and set of Virtual People $\mathcal{P}$

output: $NaiveAssociate(w, \mathcal{P})$, association consisting of Q HR sketches of which the q-th approximates the HR sketch of Virtual People represented by panelist q.

```
for q ← 1 to Q do
    s_q ← ∅
end
for p ∈ P do
    q ← PickPanelist(h(p), w)
    s_q ← s_q ∪ p
end
return GenerateHR(s_1), ..., GenerateHR(s_Q)
```

5 Fast association by deep sketch sampling

The naive association algorithm described above includes an expensive step where for each of the $Q|\mathcal{P}|$ panelist-Virutal-Person pairs a score is computed before associating the Virtual Person with the panelist with maximum pair score. To speed things up in exchange of slight accuracy degradation, we would like to avoid this $Q|\mathcal{P}|$ score computations. In other words, we want to *approximate* each panelist's sketch reasonably well without going through every panelist-Virtual-Person pair.

5.1 Algorithm

Please refer to Section 2 for a quick starter on HLL sketch and its extension HR sketch. In essence, an HR sketch consists of a predefined number of registers, and incoming objects are distributed to registers. Each object has a uniform hash, and each register stores the minimum of the uniform hashes of its incoming objects. For the purpose of our analysis we can focus on just

one register.

Since in an HR sketch each register only stores the minimum uniform hash which cardinality estimation is based upon, intuitively **only the Virtual Person that generates this minimum are important**.

Definition 1. *The deep sketch of depth D of $\mathcal{P}$ consists of the D smallest uniform hashes $S(1) \leq S(2) \leq \ldots \leq S(D)$ from $\mathcal{P}$. In other words, they are the D smallest values in $\{h(p)\}_{p \in \mathcal{P}}$ where $h(\cdot) \in (0, 1)$ is a uniform hash.*

We describe our algorithm **deep sketch sampling** in Algorithm 7. Figure 2 illustrates the algorithm.

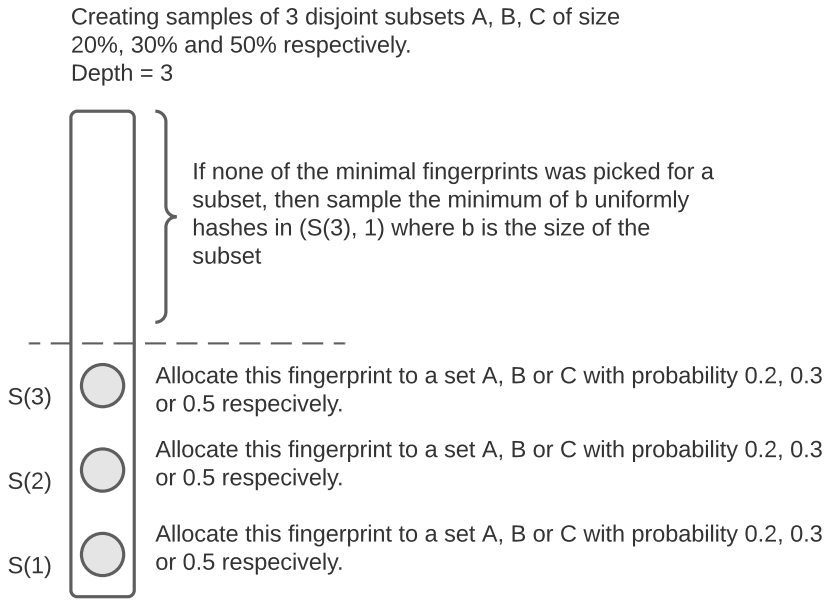

Figure 2: Deep sketch sampling. The three circles denote the deep sketch of depth 3, i.e. three smallest uniform hashes $S(1), S(2)$, and $S(3)$ drawn from Virtual People set $\mathcal{P}$

Consider a panelist q. With probability w_q this panelist gets $S(1)$ as register value and his or her sketch is finalized. Otherwise with probability

Algorithm 7: $FastAssociate(w, \mathcal{P}, D)$ produces a sketch r_q for each panelist q

input : Panelist weights $w = (w_1, \ldots, w_Q)$, set of virtual people $\mathcal{P}$, and depth D

output: $FastAssociate(w, \mathcal{P}, D)$, association consisting of Q HR sketches of which the q-th approximates the HR sketch of Virtual People represented by panelist q

$S(1), \ldots, S(D) \leftarrow$ deep sketch of depth D
$b_1, \ldots, b_Q \leftarrow MultinomialSampling(|\mathcal{P}| - D; w_1, \ldots, w_Q)$
for $q \leftarrow 1$ **to** Q **do**
  $r_q \leftarrow NULL$
end
for $d \leftarrow D$ **to** 1 **do**
  $q \leftarrow PickPanelist(S(d), w)$
  $r_q \leftarrow S(d)$
end
for $q \leftarrow 1$ **to** Q **do**
  if r_q *is NULL* **then**
    $r_q \leftarrow S(D) + (1 - S(D)) \left(1 - u^{\frac{1}{b_q}}\right)$
  end
end
return $r_1, \ldots, r_Q$

w_q they gets $S(2)$ as register value to finalize the sketch, and so on. If none of $S(1), \ldots, S(D)$ is taken by q, suppose in reality q represent b_q Virtual People out of the remaining $|\mathcal{P}| - D$. We can sample the minimum of b_q uniform random variables within $(S(D), 1)$. To see how it works, first we notice that the conditional distribution of a uniform random variable above $S(D) \leq 1$ is also uniform.

Lemma 1. *Let U be a uniform random variable and $S(D) \leq 1$. Then for any $t \in (S(D), 1)$*

$$Pr(t \leq U | S(D) \leq U) = \frac{1 - t}{1 - S(D)}.$$

With this, the conditional distribution of the minimum of b_q independent uniform random variables above $S(D) \leq 1$ has a closed form.

Corollary 1. *Let U_k be the minimum of k independent uniform random variables and $S(D) \leq 1$. Then for any $t \in (S(D), 1)$*

$$Pr(t \leq U_k | S(D) \leq U_k) = Pr\left(t \leq U | S(D) \leq U\right)^k = \left(\frac{1-t}{1-S(D)}\right)^k.$$

The first equality follows from the facts that $t \leq U_k$ implies all k uniform random variables are greater than or equal to t, $S(D) \leq U_k$ implies all k uniform random variables are greater than or equal to $S(D)$, and these k random variables are independent. By applying Inverse transform sampling[1] we arrive at the expression at Line 12 of Algorithm 7.

Finally, we perform a multinomial sampling to decide $\{b_q\}_{1 \leq q \leq Q}$, i.e. how $|\mathcal{P}| - D$ Virtual People distribute across the Q disjoint subsets.

The depth parameter D provides another way of viewing $NaiveAssociate$.

Lemma 2. *For $D \geq |\mathcal{P}|$, $FastAssociate$ and $NaiveAssociate$ produce the same output.*

Proof. The deep sketch $S(1), \ldots, S(D)$ covers the HR sketch of all Virtual People from $\mathcal{P}$, so the loop starting at Line 6 of $FastAssociate$ is equivalent to $NaiveAssociate$. Moreover because $b_q = 0$ for every q, the remaining of $FastAssociate$ after that loop does not execute. □

5.2 Deciding on depth D

It is not hard to see that $FastAssociate$ and $NaiveAssociate$ generate HR sketches with identical distribution even for $D = 0$ where we could set $S(0) = 0$. Hence if all we wanted was for the output of deep sketch sampling to provide reasonable cardinality estimate for some subset $\mathcal{P}' \subset \mathcal{P}$, then zero depth would suffice. In practice, as indicated by the **joinability requirement** in Section 3, we may have to merge it with the HR sketch for an *independent* subset $\mathcal{P}'' \subset \mathcal{P}$ produced by either normal HR generation, naive association, or even fast association. For example $\mathcal{P}$ is census, $\mathcal{P}'$ is traditional TV audience, $\mathcal{P}''$ is online audience, and without any further knowledge $\mathcal{P}'$ and $\mathcal{P}''$ are assumed to be independent. In that case we would need the merged HR sketch to give the accurate cardinality of the union of two independent sets, so the result of $FastAssociate$ should in some sense *interact* with the other HR sketch. In this subsection, we formally bound

[1] https://en.wikipedia.org/wiki/Inverse_transform_sampling

the error of the union's cardinality introduced by finite depth D, i.e. **we assume cardinality estimate is otherwise perfect**. Based on that we are then able to decide an appropriate value of depth to achieve a target error.

First, we establish a simple fact that, for the analysis in the next subsection we can treat the merged HR sketches from multiple panelists the same way as any single panelist's HR sketch.

Lemma 3. *Let q be an integer not greater than Q. Let r be obtained by merging the first q HR sketches produced by $FastAssociate(w, \mathcal{P}, D)$. Transform w to w' by merging the first q probabilities and let r' be the first sketch generated by $FastAssociate(w', \mathcal{P}, D)$. Then r and r' have the same distribution.*

Proof. Let $r_1, \ldots, r_q$ be the first q HR sketches generated by procedure $FastAssociate(w, \mathcal{P}, D)$. By definition $r = \min(r_1, \ldots, r_q)$. For a fixed $1 \leq d \leq D$, $r = S(d)$ with probability

$$
\begin{aligned}
& Pr\left(r = S(d)\right) \\
=& \sum_{i=1}^{q} Pr\left(r_i = S(d) \ \wedge \ \left(r_j > S(d) \ \forall j \neq i, \ j \leq q\right)\right) \\
=& \sum_{i=1}^{q} Pr\left(PickPanelist\left(S(d), w\right) = i \ \wedge \ \left(q < PickPanelist\left(S(d'), w\right) \ \ \forall d' < d \ \right)\right) \\
=& \sum_{i=1}^{q} w_i \left(1 - \sum_{j=1}^{q} w_j\right)^{d-1} \\
=& w'_1 \left(1 - w'_1\right)^{d-1} \\
=& Pr\left(r' = S(d)\right).
\end{aligned}
$$

When r or r' is beyond $S(D)$, they both evaluate to quantity $S(D) + (1 - S(D))\left(1 - u^{\frac{1}{b}}\right)$ where u is a uniform random variable and b is a random variable sampled from Binomial distribution $\left(|\mathcal{P}| - D, \sum_{n \leq q} w_n\right)$. □

Now, we formulate the **joinability requirement** rigorously. For each $i \in \{1, 2\}$, let $w^{(i)} = [w_1^{(i)}, \ldots, w_{Q^{(i)}}^{(i)}]$ be some panel i, and T_i be its subset. The two panels $w^{(1)}$ and $w^{(2)}$ can be disjoint, overlapping, or even identical,

but subsets T_1 and T_2 are independent.

For some $i \in \{1, 2\}$, $\mathcal{P}_i$ is the subset of $\mathcal{P}$ represented by subset T_i, i.e.

$$\{PickPanelist(h(p), w^{(i)}) : p \in \mathcal{P}_i\} = T_i,$$

and r_i is the result of running $MergeHR$ on the subset of HR sketches $FastAssociate(w^{(i)}, \mathcal{P}, D)$ corresponding to T_i. An example is that panel i is a traditional TV panel, T_i is the set of panelists who watch certain TV program, and $\mathcal{P}_i$ is the audience represented by T_i.

For $j = 3 - i$, $\mathcal{P}_j$ and HR sketch r_i could be one of the following, respectively:

- $\mathcal{P}_j$ is an independent subset of $\mathcal{P}$ similarly represented by T_j through $PickPanelist$, and r_j is similarly generated by algorithms $FastAssociate$ and $MergeHR$. For example $\mathcal{P}_j$ is another independence TV audience.
- Same as above except $FastAssociate$ is replaced by $NaiveAssociate$.
- $\mathcal{P}_j$ is an independent subset of $\mathcal{P}$, and r_j is generated by algorithms $GenerateHR$ and $MergeHR$. For instance $\mathcal{P}_j$ is an online audience that watch certain YouTube channel and is determined from YouTube log in the servers.

Regardless of the sematics of $\mathcal{P}_1$ and $\mathcal{P}_2$, they are independent and the **joinability requirement** is for the cardinality estimate of their union based on HR sketches r_1 and r_2 to be close to $|\mathcal{P}_1| + |\mathcal{P}_2| - \frac{|\mathcal{P}_1||\mathcal{P}_2|}{|\mathcal{P}|}$. If $FastAssociate$ was involved in generating both r_1 and r_2 with depth D_1 and D_2, respectively, then let D be the minimum of D_1 and D_2.

In our main result below, we bound the relative error of cardinality estimate caused by finite depth D. Contrary to convention, we normalize the error by the estimated cardinality rather than true cardinality because it is easier to bound. Numerically it matters very little if we only operate at percentage level error.

Theorem 1. *Define y and y' as the actual and estimated cardinality of $\mathcal{P}_1 \cup \mathcal{P}_2$. Then we have $\mathbb{E}\left[\frac{y'-y}{y'}\right] \leq \frac{(2D)^{2D}}{2(2D+1)^{2D+1}}$. This approaches $\frac{1}{4eD}$ as D goes to infinity.*

Proof. We reason about the value of r in the context of the deep sketch of depth D. With probability $1 - \left((1-p_1)^D(1-p_2)^D\right)$, either r_1, r_2, or both take value from the deep sketch of depth D, and therefore so does r. This translates to true expected union size $(p_1 + p_2 - p_1p_2)|\mathcal{P}|$ and relative error of zero. Otherwise, r is the minimum of $|\mathcal{P}|(p_1 + p_2)$ *independent* uniform hashes drawn from $(S(D), 1)$, which translates to estimated cardinality of $|\mathcal{P}|(p_1 + p_2)$ and relative error of $\frac{p_1+p_2-(p_1+p_2-p_1p_2)}{p_1+p_2} = \frac{p_1p_2}{p_1+p_2}$. So

$$\mathbb{E}\left[\frac{y' - y}{y'}\right] = (1-p_1)^D(1-p_2)^D\frac{p_1p_2}{p_1+p_2}.$$

With $p_1 + p_2$ fixed, it is maximized when $p_1 = p_2$, so we reduce it to

$$(1-p)^{2D}\frac{p^2}{2p} = \frac{1}{2}p(1-p)^{2D} = \frac{1}{2}(2D)^{2D}p\left(\frac{1-p}{2D}\right)^{2D} \leq \frac{(2D)^{2D}}{2(2D+1)^{2D+1}}.$$

□

Remark 1. *The cardinality estimate based on the merged HR sketch* $r = \min(r_1, r_2)$ *described above could only have an over estimate.*

We can therefore select D based on this upper bound. For example, if we want the finite depth to bring no more than 1% additional error, we could set $D = 9$ to achieve 0.94% relative error.

References

[1] Stefan Heule, Marc Nunkesser, and Alexander Hall, *Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm*, Proceedings of the 16th International Conference on Extending Database Technology, 2013, pp. 683–692.

[2] Jim Koehler, Evgeny Skvortsov, Sheng Ma, and Song Liu, *Measuring cross-device online audiences*, Tech. report, Google, Inc., 2016, available at `https://research.google/pubs/pub45353/`.

[3] Jim Koehler, Evgeny Skvortsov, and Wiesner Vos, *A method for measuring online audiences*, Tech. report, Google Inc, 2013, available at `https://research.google/pubs/pub41089/`.

[4] Frédéric Meunier, Olivier Gandouet, Éric Fusy, and Philippe Flajolet, *Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm*, Discrete Mathematics & Theoretical Computer Science (2007).

[5] Evgeny Skvortsov, *Deep liquidlegions sketch sampling*, https://colab.sandbox.google.com/github/world-federation-of-advertisers/virtual_people_examples/blob/main/notebooks/DeepLiquidSampling.ipynb, 2020.

[6] Evgeny Skvortsov and Jim Koehler, *Virtual people: Actionable reach modeling*, (2019), available at https://research.google/pubs/pub48387/.

[7] Andreas Ulbrich, Evgeny Sergeevich Skvortsov, Jeffrey Scott Wilhelm, Josh Bao, Lawrence Tsang, and Will Bradbury, *Tracking audience statistics with hyperloglog*, Tech. report, 2021, available at https://research.google/pubs/pub50207/.

[8] Wikipedia, *Exponential distribution*, https://en.wikipedia.org/wiki/Exponential_distribution#Distribution_of_the_minimum_of_exponential_random_variables.