Improved Densification of One Permutation Hashing

Anshumali Shrivastava
Department of Computer Science
Computing and Information Science
Cornell University
Ithaca, NY 14853, USA
anshu@cs.cornell.edu

Ping Li
Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
pingli@stat.rutgers.edu

Abstract

The existing work on densification of one permutation hashing [24] reduces the query processing cost of the (K, L)-parameterized Locality Sensitive Hashing (LSH) algorithm with minwise hashing, from $O(dKL)$ to merely $O(d + KL)$, where d is the number of nonzeros of the data vector, K is the number of hashes in each hash table, and L is the number of hash tables. While that is a substantial improvement, our analysis reveals that the existing densification scheme in [24] is sub-optimal. In particular, there is no enough randomness in that procedure, which affects its accuracy on very sparse datasets.

In this paper, we provide a new densification procedure which is provably better than the existing scheme [24]. This improvement is more significant for very sparse datasets which are common over the web. The improved technique has the same cost of $O(d + KL)$ for query processing, thereby making it strictly preferable over the existing procedure. Experimental evaluations on public datasets, in the task of hashing based near neighbor search, support our theoretical findings.

1 Introduction

Binary representations are common for high dimensional sparse data over the web [8, 25, 26, 1], especially for text data represented by high-order n-grams [4, 12]. Binary vectors can also be equivalently viewed as sets, over the universe of all the features, containing only locations of the non-zero entries. Given two sets S_1, $S_2 \subseteq \Omega = \{1, 2, ..., D\}$, a popular measure of similarity between sets (or binary vectors) is the *resemblance* R, defined as

$$R = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{a}{f_1 + f_2 - a}, \tag{1}$$

where $f_1 = |S_1|$, $f_2 = |S_2|$, and $a = |S_1 \cap S_2|$.

It is well-known that minwise hashing belongs to the *Locality Sensitive Hashing (LSH)* family [5, 9]. The method applies a random permutation $\pi : \Omega \to \Omega$, on the given set S, and stores the minimum value after the permutation mapping. Formally,

$$h_\pi(S) = \min(\pi(S)). \tag{2}$$

Given sets S_1 and S_2, it can be shown by elementary probability arguments that

$$Pr(h_\pi(S_1) = h_\pi(S_2)) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = R. \tag{3}$$

The probability of collision (equality of hash values), under minwise hashing, is equal to the similarity of interest R. This property, also known as the *LSH property* [14, 9], makes minwise hash functions h_π suitable for creating hash buckets, which leads to sublinear algorithms for similarity search. Because of this same LSH property, minwise hashing is a popular indexing technique for a variety of large-scale data processing applications, which include duplicate detection [4, 13], all-pair similarity [3], fast linear learning [19], temporal correlation [10], 3-way similarity & retrieval [17, 23], graph algorithms [6, 11, 21], and more.

Querying with a standard (K, L)-parameterized LSH algorithm [14], for fast similarity search, requires computing $K \times L$ min-hash values per query, where K is the number of hashes in each hash table and L is the number of hash tables. In theory, the value of KL grows with the data size [14]. In practice, typically, this number ranges from a few hundreds to a few thousands. Thus, processing a single query, for near-neighbor search, requires evaluating hundreds or thousands of independent permutations π (or cheaper universal approximations to permutations [7, 22, 20]) over the given data vector. If d denotes the number of non-zeros in the query vector, then the query preprocessing cost is $O(dKL)$ which is also the bottleneck step in the LSH algorithm [14]. Query time (latency) is crucial in many user-facing applications, such as search.

Linear learning with b-bit minwise hashing [19], requires multiple evaluations (say k) of h_π for a given data vector. Computing k different min-hashes of the test data costs $O(dk)$, while after processing, classifying this data vector

(with SVM or logistic regression) only requires a single inner product with the weight vector which is $O(k)$. Again, the bottleneck step during testing prediction is the evaluation of k min-hashes. Testing time directly translates into the latency of on-line classification systems.

The idea of storing k contiguous minimum values after one single permutation [4, 15, 16] leads to hash values which do not satisfy the LSH property because the hashes are not properly aligned. The estimators are also not linear, and therefore they do not lead to feature representation for linear learning with resemblance. This is a serious limitation.

Recently it was shown that a "rotation" technique [24] for densifying sparse sketches from one permutation hashing [18] solves the problem of costly processing with minwise hashing (See Sec. 2). The scheme only requires a single permutation and generates k different hash values, satisfying the LSH property (i.e., Eq.(3)), in linear time $O(d + k)$, thereby reducing a factor d in the processing cost compared to the original minwise hashing.

Our Contributions: In this paper, we argue that the existing densification scheme [24] is not the optimal way of densifying the sparse sketches of one permutation hashing at the given processing cost. In particular, we provide a provably better densification scheme for generating k hashes with the same processing cost of $O(d + k)$. Our contributions can be summarized as follows.

- Our detailed variance analysis of the hashes obtained from the existing densification scheme [24] reveals that there is no enough randomness in that procedure which leads to high variance in very sparse datasets.

- We provide a new densification scheme for one permutation hashing with provably smaller variance than the scheme in [24]. The improvement becomes more significant for very sparse datasets which are common in practice. The improved scheme retains the computational complexity of $O(d + k)$ for computing k different hash evaluations of a given vector.

- We provide experimental evidences on publicly available datasets, which demonstrate the superiority of the improved densification procedure over the existing scheme, in the task of resemblance estimation and as well as the task of near neighbor retrieval with LSH.

2 Background

2.1 One Permutation Hashing

As illustrated in Figure 1, instead of conducting k independent permutations, *one permutation hashing* [18] uses only one permutation and partitions the (permuted) feature space into k bins. In other words, a single permutation π is used to first shuffle the given binary vector, and then the shuffled vector is binned into k evenly spaced bins. The

k minimums, computed for each bin separately, are the k different hash values. Obviously, empty bins are possible.

	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
$\pi(\Omega)$	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23
	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3
$\pi(S_1)$	0 0 0 0	0 1 0 1	0 0 0 0	0 0 1 1	1 0 1 0	0 1 1 0
$\pi(S_2)$	0 0 0 0	0 1 1 1	0 0 0 0	1 0 1 0	1 1 0 0	0 0 0 0
OPH(S_1)	E	1	E	2	0	1
OPH(S_2)	E	1	E	0	0	E

Figure 1: One permutation hashes [18] for vectors S_1 and S_2 using a single permutation π. For bins not containing any non-zeros, we use special symbol "E".

For example, in Figure 1, $\pi(S_1)$ and $\pi(S_2)$ denote the state of the binary vectors S_1 and S_2 after applying permutation π. These shuffled vectors are then divided into 6 bins of length 4 each. We start the numbering from 0. We look into each bin and store the corresponding minimum non-zero index. For bins not containing any non-zeros, we use a special symbol "E" to denote empty bins. We also denote

$$M_j(\pi(S)) = \left\{ \pi(S) \cap \left[\frac{Dj}{k}, \frac{D(j+1)}{k} \right) \right\} \quad (4)$$

We assume for the rest of the paper that D is divisible by k, otherwise we can always pad extra dummy features. We define OPH_j ("OPH" for one permutation hashing) as

$$OPH_j(\pi(S)) = \begin{cases} E, & \text{if } \pi(S) \cap \left[\frac{Dj}{k}, \frac{D(j+1)}{k} \right) = \phi \\ M_j(\pi(S)) \bmod \frac{D}{k}, & \text{otherwise} \end{cases}$$

$$(5)$$

i.e., $OPH_j(\pi(S))$ denotes the minimum value in Bin j, under permutation mapping π, as shown in the example in Figure 1. If this intersection is null, i.e., $\pi(S) \cap \left[\frac{Dj}{k}, \frac{D(j+1)}{k} \right) = \phi$, then $OPH_j(\pi(S)) = E$.

Consider the events of "simultaneously empty bin" $I_{emp}^j = 1$ and "simultaneously non-empty bin" $I_{emp}^j = 0$, between given vectors S_1 and S_2, defined as:

$$I_{emp}^j = \begin{cases} 1, & \text{if } OPH_j(\pi(S_1)) = OPH_j(\pi(S_2)) = E \\ 0 & \text{otherwise} \end{cases}$$

$$(6)$$

Simultaneously empty bins are only defined with respect to two sets S_1 and S_2. In Figure 1, $I_{emp}^0 = 1$ and $I_{emp}^2 = 1$, while $I_{emp}^1 = I_{emp}^3 = I_{emp}^4 = I_{emp}^5 = 0$. Bin 5 is only empty for S_2 and not for S_1, so $I_{emp}^5 = 0$.

Given a bin number j, if it is not simultaneously empty

$(I_{emp}^j = 0)$ for both the vectors S_1 and S_2, [18] showed

$$Pr\left(OPH_j(\pi(S_1)) = OPH_j(\pi(S_2))\middle| I_{emp}^j = 0\right) = R \tag{7}$$

On the other hand, when $I_{emp}^j = 1$, no such guarantee exists. When $I_{emp}^j = 1$ collision does not have enough information about the similarity R. Since the event $I_{emp}^j = 1$ can only be determined given the two vectors S_1 and S_2 and the materialization of π, one permutation hashing cannot be directly used for indexing, especially when the data are very sparse. In particular, $OPH_j(\pi(S))$ does not lead to a valid LSH hash function because of the coupled event $I_{emp}^j = 1$ in (7). The simple strategy of ignoring empty bins leads to biased estimators of resemblance and shows poor performance [24]. Because of this same reason, one permutation hashing cannot be directly used to extract random features for linear learning with resemblance kernel.

2.2 Densifying One Permutation Hashing for Indexing and Linear Learning

[24] proposed a "rotation" scheme that assigns new values to all the empty bins, generated from one permutation hashing, in an unbiased fashion. The rotation scheme for filling the empty bins from Figure 1 is shown in Figure 2. The idea is that for every empty bin, the scheme borrows the value of the closest non-empty bin in the clockwise direction (circular right hand side) added with offset C.

	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
H(S₁)	1+C ←	1	2+C ←	2	0	1
H(S₂)	1+C ←	1	0+C ←	0	0	1+2C

Figure 2: Densification by "rotation" for filling empty bins generated from one permutation hashing [24]. Every empty bin is assigned the value of the closest non-empty bin, towards right (circular), with an offset C. For the configuration shown in Figure 1, the above figure shows the new assigned values (in red) of empty bins after densification.

Given the configuration in Figure 1, for Bin 2 corresponding to S_1, we borrow the value 2 from Bin 3 along with an additional offset of C. Interesting is the case of Bin 5 for S_2, the circular right is Bin 0 which was empty. Bin 0 borrows from Bin 1 acquiring value $1 + C$, Bin 5 borrows this value with another offset C. The value of Bin 5 finally becomes $1 + 2C$. The value of $C = \frac{D}{k} + 1$ enforces proper alignment and ensures no unexpected collisions. Without this offset C, Bin 5, which was not simultaneously empty, after reassignment, will have value 1 for both S_1 and S_2. This would be an error as initially there was no collision (note $I_{emp}^5 = 0$). Multiplication by the distance of the non-empty bin, from where the value was borrowed, ensures

that the new values of simultaneous empty bins ($I_{emp}^j = 1$), at any location j for S_1 and S_2, never match if their new values come from different bin numbers.

Formally the hashing scheme with "rotation", denoted by $\mathcal{H}$, is defined as:

$$\mathcal{H}_j(S) = \begin{cases} OPH_j(\pi(S)) & \text{if } OPH_j(\pi(S)) \neq E \\ OPH_{(j+t) \bmod k}(\pi(S)) + tC & \text{otherwise} \end{cases} \tag{8}$$

$$t = \min z, \quad s.t. \quad OPH_{(j+z) \bmod k}(\pi(S)) \neq E \tag{9}$$

Here $C = \frac{D}{k} + 1$ is a constant.

This densification scheme ensures that whenever $I_{emp}^j = 0$, i.e., Bin j is simultaneously empty for any two S_1 and S_2 under considerations, the newly assigned value mimics the collision probability of the nearest simultaneously non-empty bin towards right (circular) hand side making the final collision probability equal to R, irrespective of whether $I_{emp}^j = 0$ or $I_{emp}^j = 1$. [24] proved this fact as a theorem.

Theorem 1 *[24]*

$$\mathbf{Pr}\left(\mathcal{H}_j(S_1) = \mathcal{H}_j(S_2)\right) = R \tag{10}$$

Theorem 1 implies that $\mathcal{H}$ satisfies the LSH property and hence it is suitable for indexing based sublinear similarity search. Generating KL different hash values of $\mathcal{H}$ only requires $O(d + KL)$, which saves a factor of d in the query processing cost compared to the cost of $O(dKL)$ with traditional minwise hashing. For fast linear learning [19] with k different hash values the new scheme only needs $O(d+k)$ testing (or prediction) time compared to standard b-bit minwise hashing which requires $O(dk)$ time for testing.

3 Variance Analysis of Existing Scheme

We first provide the variance analysis of the existing scheme [24]. Theorem 1 leads to an unbiased estimator of R between S_1 and S_2 defined as:

$$\hat{R} = \frac{1}{k} \sum_{j=0}^{k-1} \mathbf{1}\{\mathcal{H}_j(S_1) = \mathcal{H}_j(S_2)\}. \tag{11}$$

Denote the number of simultaneously empty bins by

$$N_{emp} = \sum_{j=0}^{k-1} \mathbf{1}\{I_{emp}^j = 1\}, \tag{12}$$

where $\mathbf{1}$ is the indicator function. We partition the event $(\mathcal{H}_j(S_1) = \mathcal{H}_j(S_2))$ into two cases depending on I_{emp}^j. Let M_j^N (*Non-empty Match at j*) and M_j^E (*Empty Match at j*) be the events defined as:

$$M_j^N = \mathbf{1}\{I_{emp}^j = 0 \text{ and } \mathcal{H}_j(S_1) = \mathcal{H}_j(S_2)\} \tag{13}$$

$$M_j^E = \mathbf{1}\{I_{emp}^j = 1 \text{ and } \mathcal{H}_j(S_1) = \mathcal{H}_j(S_2)\} \tag{14}$$

Note that, $M_j^N = 1 \implies M_j^E = 0$ and $M_j^E = 1 \implies M_j^N = 0$. This combined with Theorem 1 implies,

$$\mathbb{E}(M_j^N|I_{emp}^j = 0) = \mathbb{E}(M_j^E|I_{emp}^j = 1)$$
$$= \mathbb{E}(M_j^E + M_j^N) = R \;\; \forall j \quad (15)$$

It is not difficult to show that,

$$\mathbb{E}\left(M_j^N M_i^N \middle| i \neq j, I_{emp}^j = 0 \text{ and } I_{emp}^i = 0\right) = R\tilde{R},$$

where $\tilde{R} = \frac{a-1}{f1+f2-a-1}$. Using these new events, we have

$$\hat{R} = \frac{1}{k}\sum_{j=0}^{k-1}\left[M_j^E + M_j^N\right] \quad (16)$$

We are interested in computing

$$Var(\hat{R}) = \mathbb{E}\left(\left(\frac{1}{k}\sum_{j=0}^{k-1}\left[M_j^E + M_j^N\right]\right)^2\right) - R^2 \quad (17)$$

For notational convenience we will use m to denote the event $k - N_{emp} = m$, i.e., the expression $\mathbb{E}(.|m)$ means $\mathbb{E}(.|k - N_{emp} = m)$. To simplify the analysis, we will first compute the conditional expectation

$$f(m) = \mathbb{E}\left(\left(\frac{1}{k}\sum_{j=0}^{k-1}\left[M_j^E + M_j^N\right]\right)^2 \middle| m\right) \quad (18)$$

By expansion and linearity of expectation, we obtain

$$k^2 f(m) = \mathbb{E}\left[\sum_{i\neq j} M_i^N M_j^N \middle| m\right] + \mathbb{E}\left[\sum_{i\neq j} M_i^N M_j^E \middle| m\right]$$

$$+ \mathbb{E}\left[\sum_{i\neq j} M_i^E M_j^E \middle| m\right] + \mathbb{E}\left[\sum_{i=1}^{k}\left[(M_j^N)^2 + (M_j^E)^2\right] \middle| m\right]$$

$M_j^N = (M_j^N)^2$ and $M_j^E = (M_j^E)^2$ as they are indicator functions and can only take values 0 and 1. Hence,

$$\mathbb{E}\left[\sum_{j=0}^{k-1}\left[(M_j^N)^2 + (M_j^E)^2\right] \middle| m\right] = kR \quad (19)$$

The values of the remaining three terms are given by the following 3 Lemmas; See the proofs in the Appendix.

Lemma 1

$$\mathbb{E}\left[\sum_{i\neq j} M_i^N M_j^N \middle| m\right] = m(m-1)R\tilde{R} \quad (20)$$

Lemma 2

$$\mathbb{E}\left[\sum_{i\neq j} M_i^N M_j^E \middle| m\right] = 2m(k-m)\left[\frac{R}{m} + \frac{(m-1)R\tilde{R}}{m}\right] \quad (21)$$

Lemma 3

$$\mathbb{E}\left[\sum_{i\neq j} M_i^E M_j^E \middle| m\right] = (k-m)(k-m-1)$$
$$\times \left[\frac{2R}{m+1} + \frac{(m-1)R\tilde{R}}{m+1}\right] \quad (22)$$

Combining the expressions from the above 3 Lemmas and Eq.(19), we can compute $f(m)$. Taking a further expectation over values of m to remove the conditional dependency, the variance of $\hat{R}$ can be shown in the next Theorem.

Theorem 2

$$Var(\hat{R}) = \frac{R}{k} + A\frac{R}{k} + B\frac{R\tilde{R}}{k} - R^2 \quad (23)$$
$$A = 2\mathbb{E}\left[\frac{N_{emp}}{k - N_{emp} + 1}\right]$$
$$B = (k+1)\mathbb{E}\left[\frac{k - N_{emp} - 1}{k - N_{emp} + 1}\right]$$

The theoretical values of A and B can be computed using the probability of the event $Pr(N_{emp} = i)$, denoted by P_i, which is given by Theorem 3 in [18].

$$P_i = \sum_{s=0}^{k-i}\frac{(-1)^s k!}{i!s!(k-i-s)!}\prod_{t=0}^{f_1+f_2-a-1}\frac{D\left(1 - \frac{i+s}{k}\right) - t}{D - t}$$

4 Intuition for the Improved Scheme
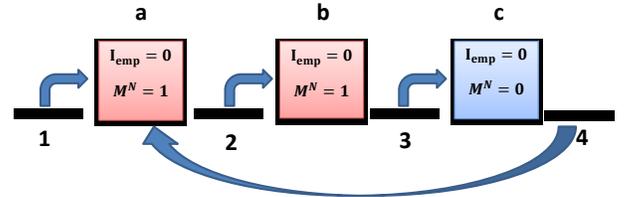
Figure 3: Illustration of the existing densification scheme [24]. The 3 boxes indicate 3 simultaneously non-empty bins. Any simultaneously empty bin has 4 possible positions shown by blank spaces. Arrow indicates the choice of simultaneous non-empty bins picked by simultaneously empty bins occurring in the corresponding positions. A simultaneously empty bin occurring in position 3 uses the information from Bin c. The randomness is in the position number of these bins which depends on π.

Consider a situation in Figure 3, where there are 3 simultaneously non-empty bins ($I_{emp} = 0$) for given S_1 and S_2. The actual position numbers of these simultaneously non-empty bins are random. The simultaneously empty bins ($I_{emp} = 1$) can occur in any order in the 4 blank spaces. The arrows in the figure show the simultaneously

non-empty bins which are being picked by the simultaneously empty bins ($I_{emp} = 1$) located in the shown blank spaces. The randomness in the system is in the ordering of simultaneously empty and simultaneously non-empty bins.

Given a simultaneously non-empty Bin t ($I^t_{emp} = 0$), the probability that it is picked by a given simultaneously empty Bin i ($I^i_{emp} = 1$) is exactly $\frac{1}{m}$. This is because the permutation π is perfectly random and given m, any ordering of m simultaneously non-empty bins and $k - m$ simultaneously empty bins are equally likely. Hence, we obtain the term $\left[\frac{R}{m} + \frac{(m-1)R\tilde{R}}{m}\right]$ in Lemma 2.

On the other hand, under the given scheme, the probability that two simultaneously empty bins, i and j, (i.e., $I^i_{emp} = 1$, $I^j_{emp} = 1$), both pick the same simultaneous non-empty Bin t ($I^t_{emp} = 0$) is given by (see proof of Lemma 3)

$$p = \frac{2}{m+1} \qquad (24)$$

The value of p is high because there is no enough randomness in the selection procedure. Since $R \leq 1$ and $R \leq R\tilde{R}$, if we can reduce this probability p then we reduce the value of $[pR + (1 - p)R\tilde{R}]$. This directly reduces the value of $(k - m)(k - m - 1)\left[\frac{2R}{m+1} + \frac{(m-1)R\tilde{R}}{m+1}\right]$ as given by Lemma 3. The reduction scales with N_{emp}.

For every simultaneously empty bin, the current scheme uses the information of the closest non-empty bin in the right. Because of the symmetry in the arguments, changing the direction to left instead of right also leads to a valid densification scheme with exactly same variance. This is where we can infuse randomness without violating the alignment necessary for unbiased densification. We show that randomly switching between left and right provably improves (reduces) the variance by making the sampling procedure of simultaneously non-empty bins more random.

5 The Improved Densification Scheme
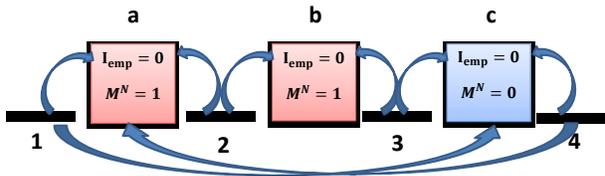
Figure 4: Illustration of the improved densification scheme. For every simultaneously empty bin, in the blank position, instead of always choosing the simultaneously non-empty bin from right, the new scheme randomly chooses to go either left or right. A simultaneously empty bin occurring at position 2 uniformly chooses among Bin a or Bin b.

Our proposal is explained in Figure 4. Instead of using the value of the closest non-empty bin from the right (circular),

we will choose to go either left or right with probability $\frac{1}{2}$. This adds more randomness in the selection procedure.

In the new scheme, we only need to store 1 random bit for each bin, which decides the direction (circular left or circular right) to proceed for finding the closest non-empty bin. The new assignment of the empty bins from Figure 1 is shown in Figure 5. Every bin number i has an i.i.d. Bernoulli random variable q_i (1 bit) associated with it. If Bin i is empty, we check the value of q_i. If $q_i = 1$, we move circular right to find the closest non-empty bin and use its value. In case when $q = 0$, we move circular left.

	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
Direction Bits (q)	0	1	0	0	1	1
H⁺(S₁)	1+C	1 → 1+C	2	0	1
H⁺(S₂)	0+2C	1 → 1+C	0	0	1+2C

Figure 5: Assigned values (in red) of empty bins from Figure 1 using the improved densification procedure. Every empty Bin i uses the value of the closest non-empty bin, towards circular left or circular right depending on the random direction bit q_i, with offset C.

For S_1, we have $q_0 = 0$ for empty Bin 0, we therefore move circular left and borrow value from Bin 5 with offset C making the final value $1 + C$. Similarly for empty Bin 2 we have $q_2 = 0$ and we use the value of Bin 1 (circular left) added with C. For S_2 and Bin 0, we have $q_0 = 0$ and the next circular left bin is Bin 5 which is empty so we continue and borrow value from Bin 4, which is 0, with offset $2C$. It is a factor of 2 because we traveled 2 bins to locate the first non-empty bin. For Bin 2, again $q_2 = 0$ and the closest circular left non-empty bin is Bin 1, at distance 1, so the new value of Bin 2 for S_2 is $1 + C$. For Bin 5, $q_5 = 1$, so we go circular right and find non-empty Bin 1 at distance 2. The new hash value of Bin 5 is therefore $1 + 2C$. Note that the non-empty bins remain unchanged.

Formally, let q_j $j = \{0, 1, 2, ..., k - 1\}$ be k i.i.d. Bernoulli random variables such that $q_j = 1$ with probability $\frac{1}{2}$. The improved hash function $\mathcal{H}^+$ is given by

$$\mathcal{H}^+_j(S) = \begin{cases} \underset{(j-t_1)\bmod k}{OPH}(\pi(S)) + t_1 C \\ \qquad \text{if } q_j = 0 \text{ and } \underset{j}{OPH}(\pi(S)) = E \\ \\ \underset{(j+t_2)\bmod k}{OPH}(\pi(S)) + t_2 C \\ \qquad \text{if } q_j = 1 \text{ and } \underset{j}{OPH}(\pi(S)) = E \\ \\ \underset{j}{OPH}(\pi(S)) \qquad\qquad \text{otherwise} \end{cases}$$

$$(25)$$

where

$$t_1 = \min z, \quad s.t. \underset{(j-z) \bmod k}{OPH} (\pi(S)) \neq E \quad (26)$$

$$t_2 = \min z, \quad s.t. \underset{(j+z) \bmod k}{OPH} (\pi(S)) \neq E \quad (27)$$

with same $C = \frac{D}{k} + 1$. Computing k hash evaluations with $\mathcal{H}^+$ requires evaluating $\pi(S)$ followed by two passes over the k bins from different directions. The total complexity of computing k hash evaluations is again $O(d + k)$ which is the same as that of the existing densification scheme. We need an additional storage of the k bits (roughly hundreds or thousands in practice) which is practically negligible.

It is not difficult to show that $\mathcal{H}^+$ satisfies the LSH property for resemblance, which we state as a theorem.

Theorem 3

$$\mathbf{Pr}\left(\mathcal{H}_j^+(S_1) = \mathcal{H}_j^+(S_2)\right) = R \quad (28)$$

$\mathcal{H}^+$ leads to an unbiased estimator of resemblance $\hat{R}^+$

$$\hat{R}^+ = \frac{1}{k} \sum_{j=0}^{k-1} \mathbf{1}\{\mathcal{H}_j^+(S_1) = \mathcal{H}_j^+(S_2)\}. \quad (29)$$

6 Variance Analysis of Improved Scheme

When $m = 1$ (an event with prob $\left(\frac{1}{k}\right)^{f_1+f_2-a} \simeq 0$), i.e., only one simultaneously non-empty bin, both the schemes are exactly same. For simplicity of expressions, we will assume that the number of simultaneous non-empty bins is strictly greater than 1, i.e., $m > 1$. The general case has an extra term for $m = 1$, which makes the expression unnecessarily complicated without changing the final conclusion.

Following the notation as in Sec. 3, we denote

$$M_j^{N+} = \mathbf{1}\{I_{emp}^j = 0 \text{ and } \mathcal{H}_j^+(S_1) = \mathcal{H}_j^+(S_2)\} \quad (30)$$

$$M_j^{E+} = \mathbf{1}\{I_{emp}^j = 1 \text{ and } \mathcal{H}_j^+(S_1) = \mathcal{H}_j^+(S_2)\} \quad (31)$$

The two expectations $\mathbb{E}\left[\sum_{i \neq j} M_i^{N+} M_j^{N+} \middle| m\right]$ and $\mathbb{E}\left[\sum_{i \neq j} M_i^{N+} M_j^{E+} \middle| m\right]$ are the same as given by Lemma 1 and Lemma 2 respectively, as all the arguments used to prove them still hold for the new scheme. The only change is in the term $\mathbb{E}\left[\sum_{i \neq j} M_i^E M_j^E \middle| m\right]$.

Lemma 4

$$\mathbb{E}\left[\sum_{i \neq j} M_i^{E+} M_j^{E+} \middle| m\right] = (k-m)(k-m-1)$$

$$\times \left[\frac{3R}{2(m+1)} + \frac{(2m-1)R\tilde{R}}{2(m+1)}\right] \quad (32)$$

The theoretical variance of the new estimator $\hat{R}^+$ is given by the following Theorem 4.

Theorem 4

$$Var(\hat{R}^+) = \frac{R}{k} + A^+ \frac{R}{k^2} + B^+ \frac{R\tilde{R}}{k^2} - R^2 \quad (33)$$

$$A^+ = \mathbb{E}\left[\frac{N_{emp}(4k - N_{emp} + 1)}{2(k - N_{emp} + 1)}\right]$$

$$B^+ = \mathbb{E}\left[\frac{2k^3 + N_{emp}^2 - N_{emp}(2k^2 + 2k + 1) - 2k}{2(k - N_{emp} + 1)}\right]$$

The new scheme reduces the value of p (see Eq.(24)) from $\frac{2}{m+1}$ to $\frac{1.5}{m+1}$. As argued in Sec. 4, this reduces the overall variance. Here, we state it as theorem that $Var(\hat{R}^+) \leq Var(\hat{R})$ always.

Theorem 5

$$Var(\hat{R}^+) \leq Var(\hat{R}) \quad (34)$$

More precisely,

$$Var(\hat{R}) - Var(\hat{R}^+)$$
$$= \mathbb{E}\left[\frac{(N_{emp})(N_{emp} - 1)}{2k^2(k - N_{emp} + 1)}[R - R\tilde{R}]\right] \quad (35)$$

The probability of simultaneously empty bins increases with increasing sparsity in dataset and the total number of bins k. We can see from Theorem 5 that with more simultaneously empty bins, i.e., higher N_{emp}, the gain with the improved scheme $\mathcal{H}^+$ is higher compared to $\mathcal{H}$. Hence, $\mathcal{H}^+$ should be significantly better than the existing scheme for very sparse datasets or in scenarios when we need a large number of hash values.

7 Evaluations

Our first experiment concerns the validation of the theoretical variances of the two densification schemes. The second experiment focuses on comparing the two schemes in the context of near neighbor search with LSH.

7.1 Comparisons of Mean Square Errors

We empirically verify the theoretical variances of $\mathcal{R}$ and $\mathcal{R}^+$ and their effects in many practical scenarios. To achieve this, we extracted 12 pairs of words (which cover a wide spectrum of sparsity and similarity) from the web-crawl dataset which consists of word representation from 2^{16} documents. Every word is represented as a binary vector (or set) of $D = 2^{16}$ dimension, with a feature value of 1 indicating the presence of that word in the corresponding document. See Table 1 for detailed information of the data.

For all 12 pairs of words, we estimate the resemblance using the two estimators $\mathcal{R}$ and $\mathcal{R}^+$. We plot the empirical
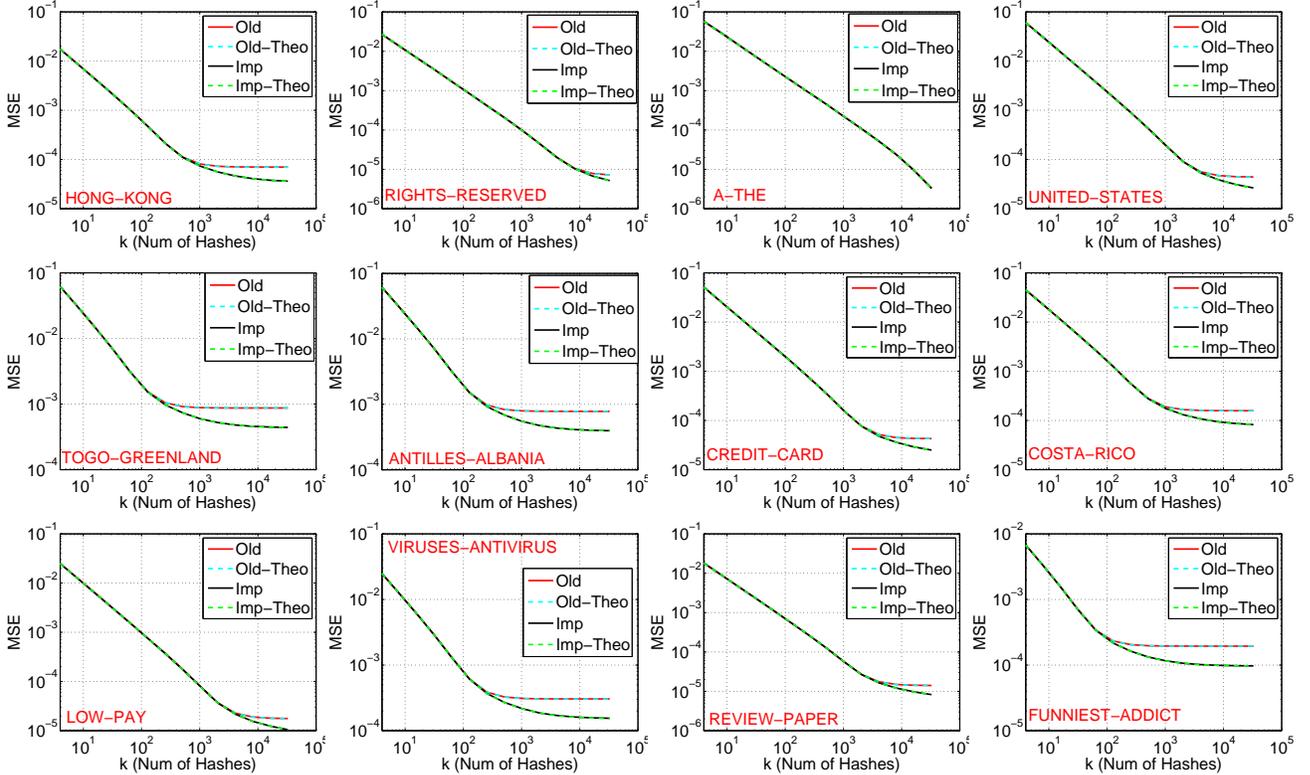
Figure 6: Mean Square Error (MSE) of the old scheme $\hat{R}$ and the improved scheme $\hat{R}^+$ along with their theoretical values on 12 word pairs (Table 1) from a web crawl dataset.

Table 1: Information of 12 pairs of word vectors. Each word stands for a set of documents in which the word is contained. For example, "A" corresponds to the set of document IDs which contained word "A".

Word 1	Word 2	f_1	f_2	R
HONG	KONG	940	948	0.925
RIGHTS	RESERVED	12,234	11,272	0.877
A	THE	39,063	42,754	0.644
UNITED	STATES	4,079	3,981	0.591
TOGO	GREENLAND	231	200	0.528
ANTILLES	ALBANIA	184	275	0.457
CREDIT	CARD	2,999	2,697	0.285
COSTA	RICO	773	611	0.234
LOW	PAY	2,936	2,828	0.112
VIRUSES	ANTIVIRUS	212	152	0.113
REVIEW	PAPER	3,197	1,944	0.078
FUNNIEST	ADDICT	68	77	0.028

Mean Square Error (MSE) of both estimators with respect to k which is the number of hash evaluations. To validate the theoretical variances (which is also the MSE because the estimators are unbiased), we also plot the values of the theoretical variances computed from Theorem 2 and Theorem 4. The results are summarized in Figure 6.

From the plots we can see that the theoretical and the empirical MSE values overlap in both the cases validating both Theorem 2 and Theorem 4. When k is small both the schemes have similar variances, but when k increases

the improved scheme always shows better variance. For very sparse pairs, we start seeing a significant difference in variance even for k as small as 100. For a sparse pair, e.g., "TOGO" and "GREENLAND", the difference in variance, between the two schemes, is more compared to the dense pair "A" and "THE". This is in agreement with Theorem 5.

7.2 Near Neighbor Retrieval with LSH

In this experiment, we evaluate the two hashing schemes $\mathcal{H}$ and $\mathcal{H}^+$ on the standard (K, L)-parameterized LSH algorithm [14, 2] for retrieving near neighbors. Two publicly available sparse text datasets are described in Table 2.

Table 2: Dataset information.

Data	# dim	# nonzeros	# train	# query
RCV1	47,236	73	100,000	5,000
URL	3,231,961	115	90,000	5,000

In (K, L)-parameterized LSH algorithm for near neighbor search, we generate L different meta-hash functions. Each of these meta-hash functions is formed by concatenating K different hash values as

$$B_j(S) = [h_{j1}(S); h_{j2}(S); ...; h_{jK}(S)], \quad (36)$$

where $h_{ij}, i \in \{1, 2, ..., K\}, j \in \{1, 2, ..., L\}$, are KL realizations of the hash function under consideration. The (K, L)-parameterized LSH works in two phases:

1. **Preprocessing Phase:** We construct L hash tables from the data by storing element S, in the train set, at location $B_j(S)$ in hash-table j.

2. **Query Phase:** Given a query Q, we report the union of all the points in the buckets $B_j(Q)$ $\forall j \in \{1, 2, ..., L\}$, where the union is over L hash tables.

For every dataset, based on the similarity levels, we chose a K based on standard recommendation. For this K we show results for a set of values of L depending on the recall values. Please refer to [2] for details on the implementation of LSH. Since both $\mathcal{H}$ and $\mathcal{H}^+$ have the same collision probability, the choice of K and L is the same in both cases.

For every query point, the gold standard top 10 near neighbors from the training set are computed based on actual resemblance. We then compute the recall of these gold standard neighbors and the total number of points retrieved by the (K, L) bucketing scheme. We report the mean computed over all the points in the query set. Since the experiments involve randomization, the final results presented are averaged over 10 independent runs. The recalls and the points retrieved per query are summarized in Figure 7.
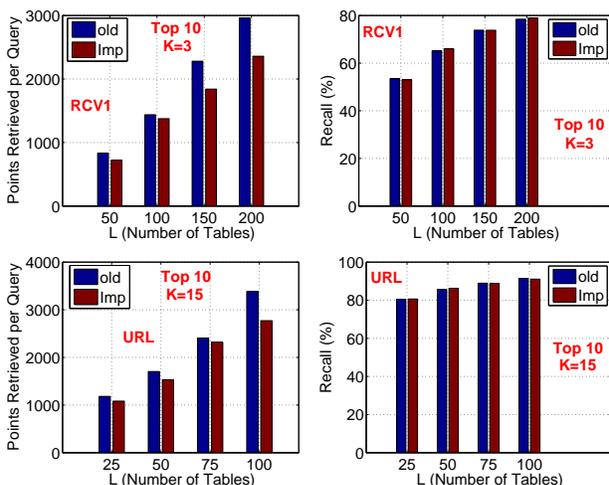
Figure 7: Average number of points scanned per query and the mean recall values of top 10 near neighbors, obtained from (K, L)-parameterized LSH algorithm, using $\mathcal{H}$ (old) and $\mathcal{H}^+$ (Imp). Both schemes achieve the same recall but $\mathcal{H}^+$ reports fewer points compared to $\mathcal{H}$. Results are averaged over 10 independent runs.

It is clear from Figure 7 that the improved hashing scheme $\mathcal{H}^+$ achieves the same recall but at the same time retrieves less number of points compared to the old scheme $\mathcal{H}$. To achieve 90% recall on URL dataset, the old scheme retrieves around 3300 points per query on an average while the improved scheme only needs to check around 2700 points per query. For RCV1 dataset, with $L = 200$ the old scheme retrieves around 3000 points and achieves a re-

call of 80%, while the same recall is achieved by the improved scheme after retrieving only about 2350 points per query. A good hash function provides a right balance between recall and number of points retrieved. In particular, a hash function which achieves a given recall and at the same time retrieves less number of points is desirable because it implies better precision. The above results clearly demonstrate the superiority of the indexing scheme with improved hash function $\mathcal{H}^+$ over the indexing scheme with $\mathcal{H}$.

7.3 Why $\mathcal{H}^+$ retrieves less number of points than $\mathcal{H}$?

The number of points retrieved, by the (K, L) parameterized LSH algorithm, is directly related to the collision probability of the meta-hash function $B_j(.)$ (Eq.(36)). Given S_1 and S_2 with resemblance R, the higher the probability of event $B_j(S_1) = B_j(S_2)$, under a hashing scheme, the more number of points will be retrieved per table.

The analysis of the variance (second moment) about the event $B_j(S_1) = B_j(S_2)$ under $\mathcal{H}^+$ and $\mathcal{H}$ provides some reasonable insight. Recall that since both estimators under the two hashing schemes are unbiased, the analysis of the first moment does not provide information in this regard.

$$\mathbb{E}\big[1\{\mathcal{H}_{j1}(S_1) = \mathcal{H}_{j1}(S_2)\} \times 1\{\mathcal{H}_{j2}(S_1) = \mathcal{H}_{j2}(S_2)\}\big]$$
$$= \mathbb{E}\big[M_{j1}^N M_{j2}^N + M_{j1}^N M_{j2}^E + M_{j1}^E M_{j2}^N + M_{j1}^E M_{j2}^E\big]$$

As we know from our analysis that the first three terms inside expectation, in the RHS of the above equation, behaves similarly for both $\mathcal{H}^+$ and $\mathcal{H}$. The fourth term $\mathbb{E}\big[M_{j1}^E M_{j2}^E\big]$ is likely to be smaller in case of $\mathcal{H}^+$ because of smaller values of p. We therefore see that $\mathcal{H}$ retrieves more points than necessary as compared to $\mathcal{H}^+$. The difference is visible when empty bins dominate and $M_1^E M_2^E = 1$ is more likely. This happens in the case of sparse datasets which are common in practice.

8 Conclusion

Analysis of the densification scheme for one permutation hashing, which reduces the processing time of minwise hashes, reveals a sub-optimality in the existing procedure. We provide a simple improved procedure which adds more randomness in the current densification technique leading to a provably better scheme, especially for very sparse datasets. The improvement comes without any compromise with the computation and only requires $O(d+k)$ (linear) cost for generating k hash evaluations. We hope that our improved scheme will be adopted in practice.

Acknowledgement

A Proofs

For the analysis, it is sufficient to consider the configurations, of empty and non-empty bins, arising after throwing $|S_1 \cup S_2|$ balls uniformly into k bins with exactly m non-empty bins and $k-m$ empty bins. Under uniform throwing of balls, any ordering of m non-empty and $k-m$ empty bins is equally likely. The proofs involve elementary combinatorial arguments of counting configurations.

A.1 Proof of Lemma 1

Given exactly m simultaneously non-empty bins, any two of them can be chosen in $m(m-1)$ ways (with ordering of i and j). Each term $M_i^N M_j^N$, for both simultaneously non-empty i and j, is 1 with probability $R\tilde{R}$ (Note, $\mathbb{E}\left(M_i^N M_j^N \big| i \neq j, I_{emp}^i = 0, I_{emp}^j = 0\right) = R\tilde{R}$).

A.2 Proof of Lemma 2

The permutation is random and any sequence of simultaneously m non-empty and remaining $k-m$ empty bins are equal likely. This is because, while randomly throwing $|S_1 \cup S_2|$ balls into k bins with exactly m non-empty bins every sequence of simultaneously empty and non-empty bins has equal probability. Given m, there are total $2m(k-m)$ different pairs of empty and non-empty bins (including the ordering). Now, for every simultaneously empty bin j, i.e., $I_{emp}^j = 1$, M_j^E replicates M_t^N corresponding to nearest non-empty Bin t which is towards the circular right. There are two cases we need to consider:

Case 1: $t = i$, which has probability $\frac{1}{m}$ and

$$\mathbb{E}(M_i^N M_j^E | I_{emp}^i = 0, I_{emp}^j = 1) = \mathbb{E}(M_i^N | I_{emp}^i = 0) = R$$

Case 2: $t \neq i$, which has probability $\frac{m-1}{m}$ and

$$\begin{aligned}&\mathbb{E}(M_i^N M_j^E | I_{emp}^i = 0, I_{emp}^j = 1)\\ =&\mathbb{E}(M_i^N M_t^N | t \neq i, I_{emp}^i = 0, \ I_{emp}^t = 0) = R\tilde{R}\end{aligned}$$

Thus, the value of $\mathbb{E}\left[\sum_{i \neq j} M_i^N M_j^E \Big| m\right]$ comes out to be

$$2m(k-m)\left[\frac{R}{m} + \frac{(m-1)R\tilde{R}}{m}\right]$$

which is the desired expression.

A.3 Proof of Lemma 3

Given m, we have $(k-m)(k-m-1)$ different pairs of simultaneous non-empty bins. There are two cases, if the closest simultaneous non-empty bins towards their circular right are identical, then for such i and j, $M_i^E M_j^E = 1$ with probability R, else $M_i^E M_j^E = 1$ with probability $R\tilde{R}$. Let p be the probability that two simultaneously empty

bins i and j have the same closest bin on the right. Then $\mathbb{E}\left[\sum_{i \neq j} M_i^E M_j^E \Big| m\right]$ is given by

$$(k-m)(k-m-1)\left[pR + (1-p)R\tilde{R}\right] \quad (37)$$

because with probability $(1-p)$, it uses estimators from different simultaneous non-empty bins and in that case the $M_i^E M_j^E = 1$ with probability $R\tilde{R}$.

Consider Figure 3, where we have 3 simultaneous non-empty bins, i.e., $m = 3$ (shown by colored boxes). Given any two simultaneous empty bins Bin i and Bin j (out of total $k - m$) they will occupy any of the $m + 1 = 4$ blank positions. The arrow shows the chosen non-empty bins for filling the empty bins. There are $(m+1)^2 + (m+1) = (m+1)(m+2)$ different ways of fitting two simultaneous non-empty bins i and j between m non-empty bins. Note, if both i and j go to the same blank position they can be permuted. This adds extra term $(m+1)$.

If both i and j choose the same blank space or the first and the last blank space, then both the simultaneous empty bins, Bin i and Bin j, corresponds to the same non-empty bin. The number of ways in which this happens is $2(m+1) + 2 = 2(m+2)$. So, we have

$$p = \frac{2(m+2)}{(m+1)(m+2)} = \frac{2}{m+1}.$$

Substituting p in Eq.(37) leads to the desired expression.

A.4 Proof of Lemma 4

Similar to the proof of Lemma 3, we need to compute p which is the probability that two simultaneously empty bins, Bin i and Bin j, use information from the same bin. As argued before, the total number of positions for any two simultaneously empty bins i and j, given m simultaneously non-empty bins is $(m+1)(m+2)$. Consider Figure 4, under the improved scheme, if both Bin i and Bin j choose the same blank position then they choose the same simultaneously non-empty bin with probability $\frac{1}{2}$. If Bin i and Bin j choose consecutive positions (e.g., position 2 and position 3) then they choose the same simultaneously non-empty bin (Bin b) with probability $\frac{1}{4}$. There are several boundary cases to consider too. Accumulating the terms leads to

$$p = \frac{\frac{2(m+2)}{2} + \frac{2m+4}{4}}{(m+1)(m+2)} = \frac{1.5}{m+1}.$$

Substituting p in Eq.(37) yields the desired result.

Note that $m = 1$ (an event with almost zero probability) leads to the value of $p = 1$. We ignore this case because it unnecessarily complicates the final expressions. $m = 1$ can be easily handled and does not affect the final conclusion.

References

[1] A. Agarwal, O. Chapelle, M. Dudik, and J. Langford. A reliable effective terascale linear learning system. Technical report, arXiv:1110.4198, 2011.

[2] A. Andoni and P. Indyk. E2lsh: Exact euclidean locality sensitive hashing. Technical report, 2004.

[3] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

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

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, pages 327–336, Dallas, TX, 1998.

[6] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM*, pages 95–106, Stanford, CA, 2008.

[7] J. L. Carter and M. N. Wegman. Universal classes of hash functions. In *STOC*, pages 106–112, 1977.

[8] T. Chandra, E. Ie, K. Goldman, T. L. Llinares, J. Mc-Fadden, F. Pereira, J. Redstone, T. Shaked, and Y. Singer. Sibyl: a system for large scale machine learning.

[9] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, Montreal, Quebec, Canada, 2002.

[10] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW*, pages 2–11, 2005.

[11] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *KDD*, pages 219–228, Paris, France, 2009.

[12] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. In *WWW*, pages 669–678, Budapest, Hungary, 2003.

[13] M. R. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR*, pages 284–291, 2006.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, Dallas, TX, 1998.

[15] P. Li and K. W. Church. Using sketches to estimate associations. In *HLT/EMNLP*, pages 708–715, Vancouver, BC, Canada, 2005.

[16] P. Li, K. W. Church, and T. J. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In *NIPS*, pages 873–880, Vancouver, BC, Canada, 2006.

[17] P. Li, A. C. König, and W. Gui. b-bit minwise hashing for estimating three-way similarities. In *Advances in Neural Information Processing Systems*, Vancouver, BC, 2010.

[18] P. Li, A. B. Owen, and C.-H. Zhang. One permutation hashing. In *NIPS*, Lake Tahoe, NV, 2012.

[19] P. Li, A. Shrivastava, J. Moore, and A. C. König. Hashing algorithms for large-scale learning. In *NIPS*, Granada, Spain, 2011.

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

[21] M. Najork, S. Gollapudi, and R. Panigrahy. Less is more: sampling the neighborhood graph makes salsa better and faster. In *WSDM*, pages 242–251, Barcelona, Spain, 2009.

[22] N. Nisan. Pseudorandom generators for space-bounded computations. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC, pages 204–212, 1990.

[23] A. Shrivastava and P. Li. Beyond pairwise: Provably fast algorithms for approximate k-way similarity search. In *NIPS*, Lake Tahoe, NV, 2013.

[24] A. Shrivastava and P. Li. Densifying one permutation hashing via rotation for fast near neighbor search. In *ICML*, Beijing, China, 2014.

[25] S. Tong. Lessons learned developing a practical large scale machine learning system. http://googleresearch.blogspot.com/2010/04/lessons-learned-developing-practical.html, 2008.

[26] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120, 2009.