Densifying One Permutation Hashing via Rotation For Fast Near-Neighbor Search

Anshumali Shrivastava and Ping Li

Cornell University and Rutgers University

June 24th 2014

Quick Summary of the Talk

We provide an efficient replacement for standard min-wise hashing.

Gains:

- Faster Sub-Linear Near Neighbor Search via Indexing (Main Focus)
- Faster Kernel Features for Linear Learning with Resemblance Kernel.
- Faster Sketching for Linear Similarity Estimation.

Sparse Binary High Dimensional Data Everywhere

- Wide adoption of the "Bag of Words" (BoW) representations for documents and images.
- Usually, when using **higher shingles**, most of the shingles only occur **at most once**.
- Most information in the sparsity structure rather than the magnitude.
- Modern "Big data" systems use binary data matrix $n \times D$, with both n and D easily running into **billions, trillions, or even** 2^{64} (e.g SIBYL).
- Often the data is extremely sparse with only few hundreds or thousands non-zeros.

The Resemblance Similarity

A binary (0/1) vector $\Longleftrightarrow$ a set (locations of nonzeros).

Consider two sets $S_1, S_2 \subseteq \Omega = \{0, 1, 2, ..., D-1\}$ (e.g., $D = 2^{64}$)
$f_1 = |S_1|, \quad f_2 = |S_2|, \quad a = |S_1 \cap S_2|.$

The **resemblance** R is a popular measure of set similarity

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

Minwise Hashing

The standard practice in the search industry: Suppose a random permutation π is performed on Ω, i.e.,

$$\pi: \ \Omega \longrightarrow \Omega, \qquad \text{where} \quad \Omega = \{0, 1, ..., D-1\}.$$

An elementary probability argument shows that

$$\mathbf{Pr}\left(\min(\pi(S_1)) = \min(\pi(S_2))\right) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = R.$$

Linear estimator: just count collisions, which is also an inner product .

Also known as LSH Property, useful for indexing.

Example: Minwise Hashing in 0/1 Data Matrix

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1:	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0
S_2:	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
S_3:	0	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(S_1)$:	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0
$\pi(S_2)$:	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
$\pi(S_3)$:	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0

Figure: Data matrix before (**Left**) and after (**Right**) permutation under π

$$\min(\pi(S_1)) = 2, \quad \min(\pi(S_2)) = 0, \quad \min(\pi(S_3)) = 0$$

Indexing Scheme for Sub-linear Retrieval

Index		Data Points
00	00	***8***, 13, 251
00	01	5, 14, 19, 29
00	10	(empty)
11	01	7, 24, 156
11	10	33, 174, 3153
11	11	61, 342

Index		Data Points
00	00	2, 19, 83
00	01	17, 36, 129
00	10	4, 34, 52, 796
11	01	7, 198
11	10	56, 989
11	11	***8***,9, 156, 879

Figure: An example of $L = 2$ hash tables, here $K = 2$ and $D = 4$.

- Given query q, if $h_1(q)$[1] $= 11$, $h_2(q) = 01$, $h_3(q) = 00$ and $h_4(q) = 01$, then probe buckets with index **1101** in the first table and **0001** in the second.
- (LSH Property) $h_i(q) = h_i(x)$ is an indicator of high resemblance between q and x for all i.
- Requires KL different hash functions $\{h_1, h_2, ..., h_{KL}\}$.

[1] $h_i(.) = \min(\pi_i(.))$

Query Bottleneck: Costly Processing

- Every min-wise hash computation requires processing the entire vector under a permutation mapping and computing the minimum $O(d)$.
- The total query complexity of minwise hashing based indexing is the cost of computing all the hash functions $O(dKL)$ and the re-ranking cost which in theory is $\leq O(dL)$. Dominated by the processing cost !!
- After shuffling the whole data vector, simply storing the minimum ought to be very wasteful.

In theory, we need $K = O(\log n)$ and $L = O(n^{\rho})$

d denotes the number of non-zeros

k-minimums do not help

- Conditional Random Sampling (CRS) (Li and Church 2005)[2] store k minimum values instead of 1.
- Estimators are not linear, cannot be used for indexing and linear learning.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(S_1)$:	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0
$\pi(S_2)$:	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0
$\pi(S_3)$:	1	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0

Min-3(S_1) $=\{2, 4, 7\}$, Min-3(S_2) $=\{0, 3, 6\}$, and Min-3(S_2) $=\{0, 1, 3\}$

The probability of $i^{th}min(\pi(S_1)) = i^{th}min(\pi(S_2))$ is not purely an indicator of high similarity for $i > 1$. (Not an LSH)

[2] CRS improved the estimator in Border et. al. 1997 and is not limited to binary data

One Permutation Hashing

Idea: Bin the data vector and use minimum within each bins as hash values.

	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
$\pi(\Omega)$	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21 22 23 24
	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4
$\pi(S_1)$	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
SKETCH(S_1)	E	2	E	3	1	2
SKETCH(S_2)	E	2	E	1	1	E

Sketches are **aligned** and hence comparable.

If a given bin is not empty for both S_1 and S_2, then
$Pr(Bin_i \min(\pi(S_1)) = Bin_i \min(\pi(S_2))) = R$ just like minwise hashing!!

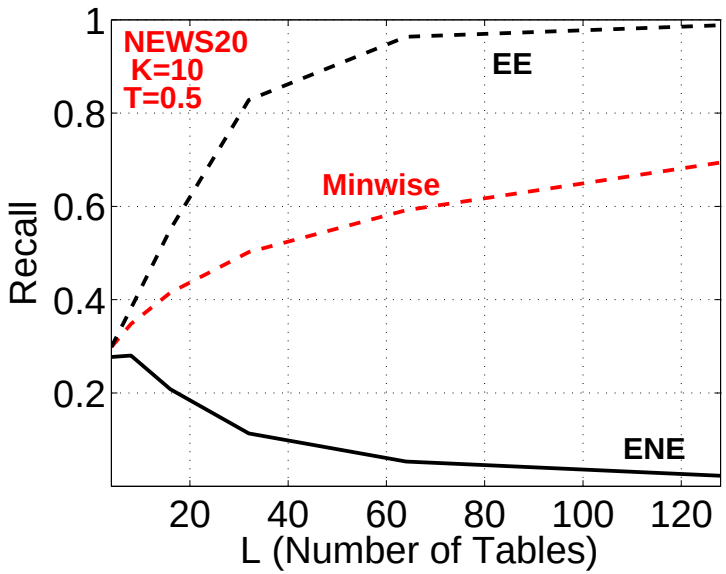
The Problem of Empty Bins

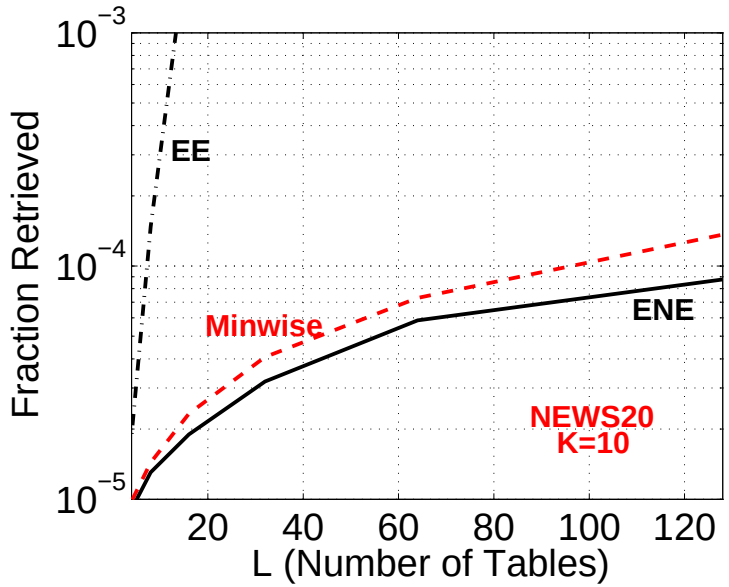
Simultaneous Empty Bins ??

- If Bin_j is empty for both vectors, it does not indicate high similarity (it may be just sparsity).
- Simultaneous empty bins is a coupled event, cannot be determined without knowing both S_1 and S_2 under consideration.
- Given any bin, we can expect lot of elements to be empty on it. (sparsity and large n).
- We don't know the query in advance. Cannot determine which bin to use !!

Does it matter?: Two Heuristics

Empty Equal Heuristic(EE): Simultaneous empty bin is an indicator of similarity. Implementation: Assign fixed special number to empty bins.
Empty Not Equal Heuristic(ENE): Simultaneous empty bin is not an indicator of similarity. Implementation: Assign new random number for every empty bin.





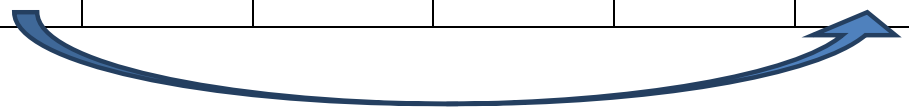
It matters!!. We need something better.

Our Proposal: Densify via Rotation

Problem: Simultaneous empty bins do not have enough information, for indexing and linear learning we need a hash value from every bin.

Solution: Empty bins borrow values from nearest non-empty bins (in clockwise direction) with some offset distance C. C ensures alignment.

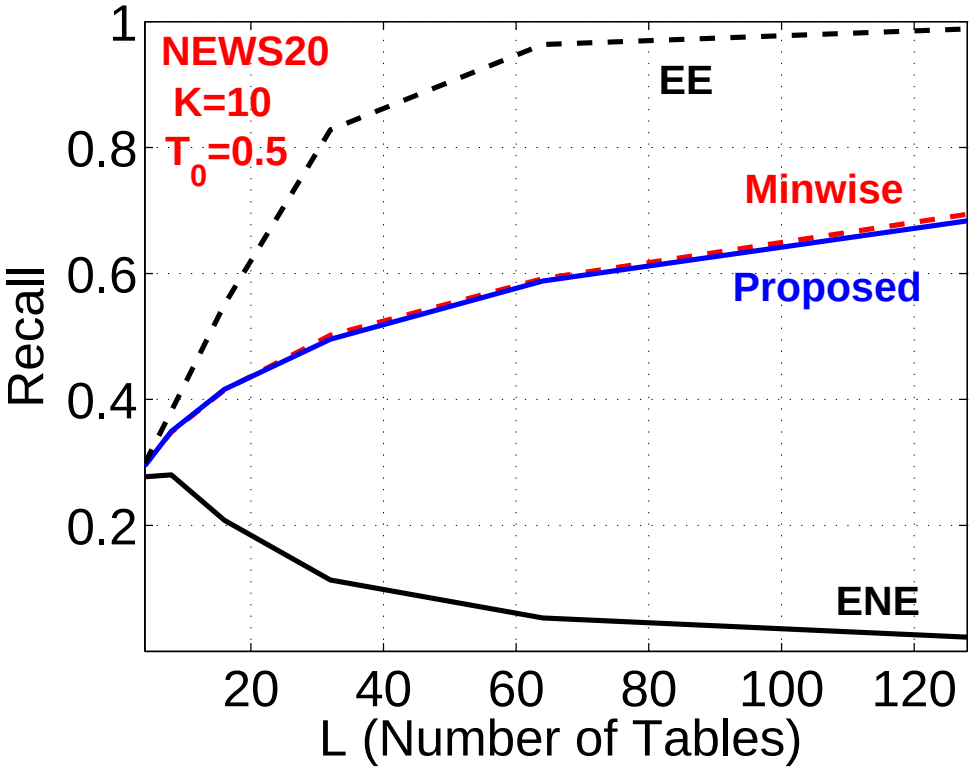
	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
$H(S_1)$	1+C ←	1	2+C ←	2	0	1
$H(S_2)$	1+C ←	1	0+C ←	0	0	1+2C

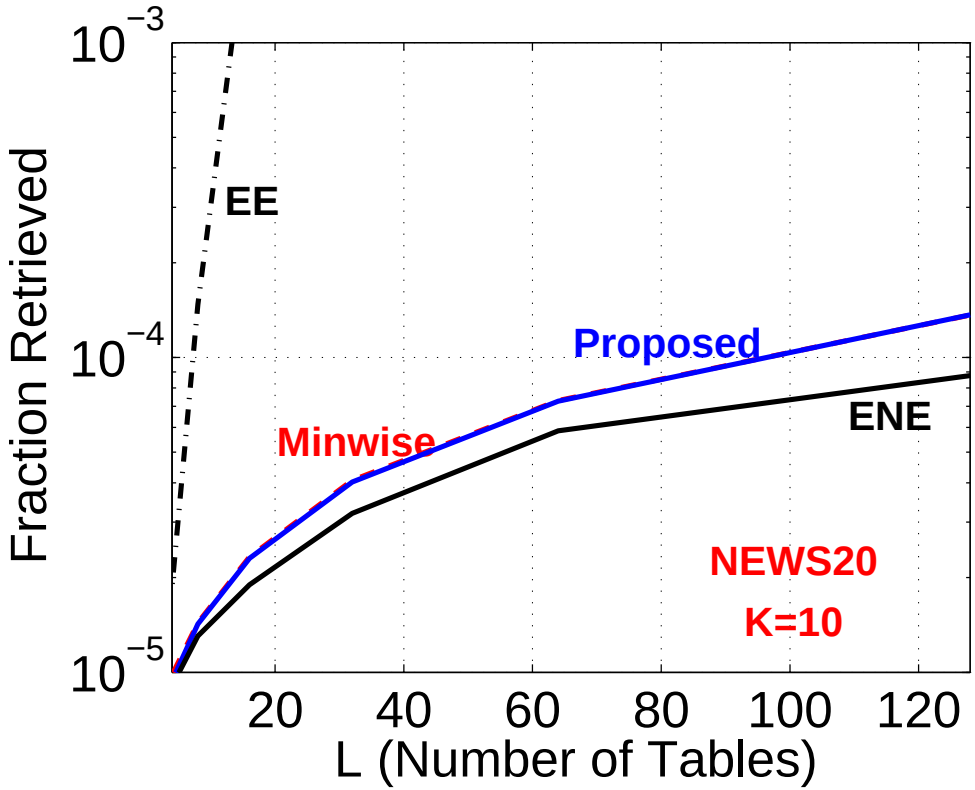


Implicitly resampling information from non-empy bins everytime we encounter simultanoues empty bins.
After reassignment, collision probability is R. Satisfies LSH property!!

Can be done in $O(d + k)$

Performance: NEWS20 Data

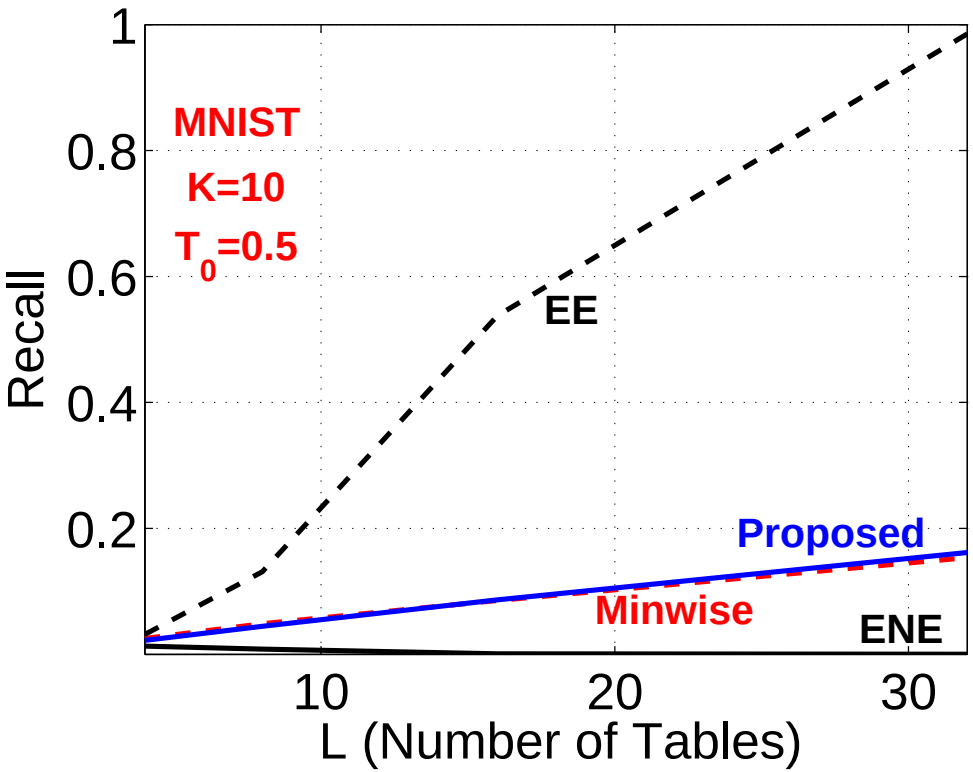


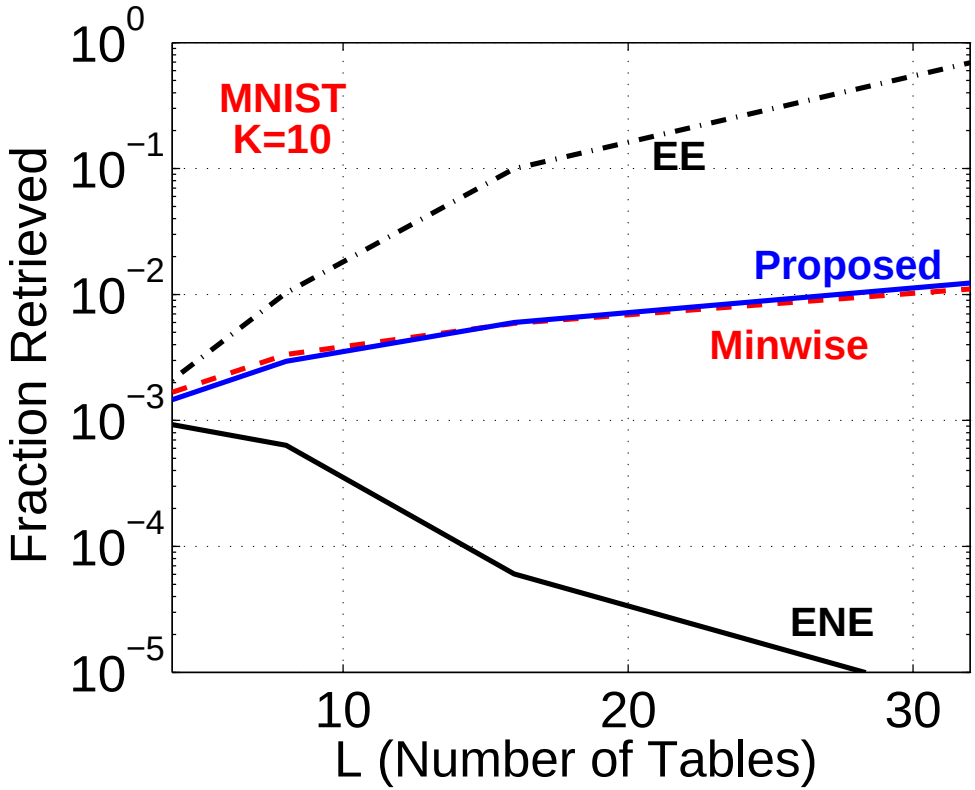


Left: Recall of points with similarity greater than 0.5.

Right: Fraction of total number of points retrieved by the algorithm.

Performance: MNIST Data

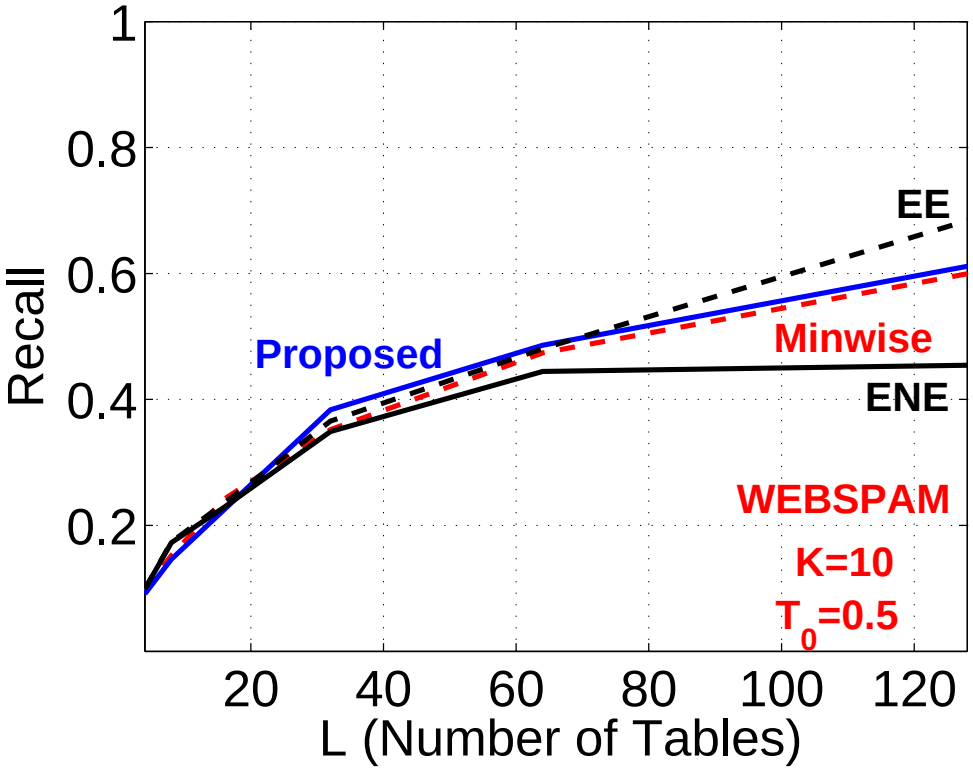


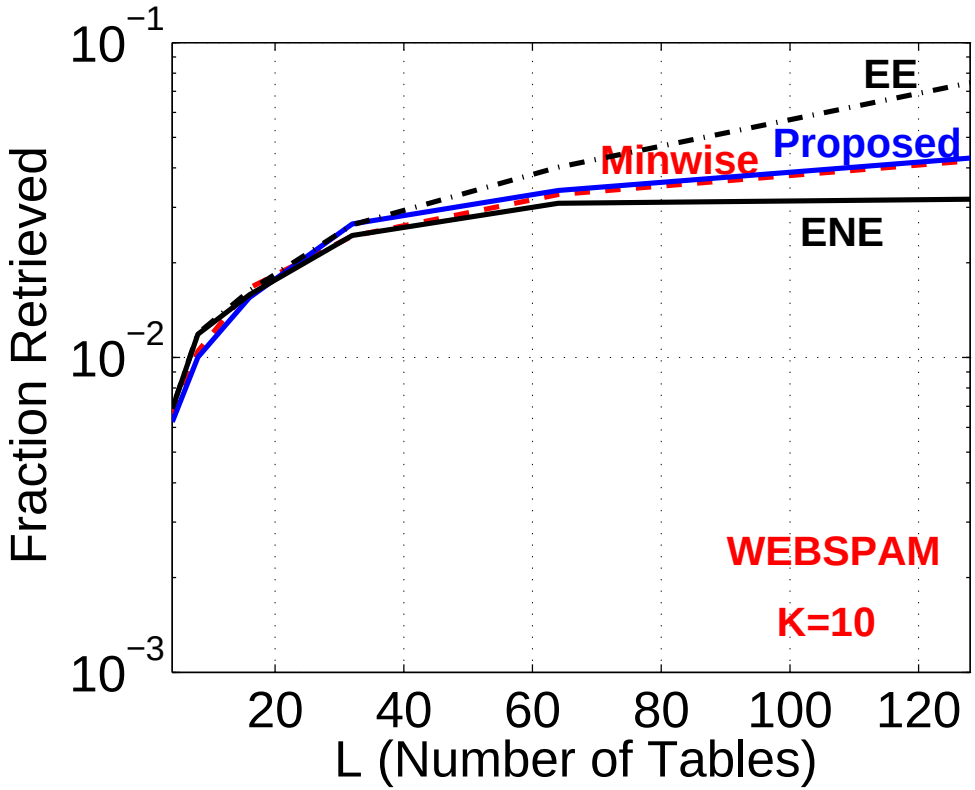


Left: Recall of points with similarity greater than 0.5.

Right: Fraction of total number of points retrieved by the algorithm.

Performance: WEBSPAM Data





Left: Recall of points with similarity greater than 0.5.

Right: Fraction of total number of points retrieved by the algorithm.

Computational Savings

Computing $K \times L$ hash evaluations requires processing one permutation followed by rotation $O(d + KL)$, compare it to $O(dKL)$ with minhash.

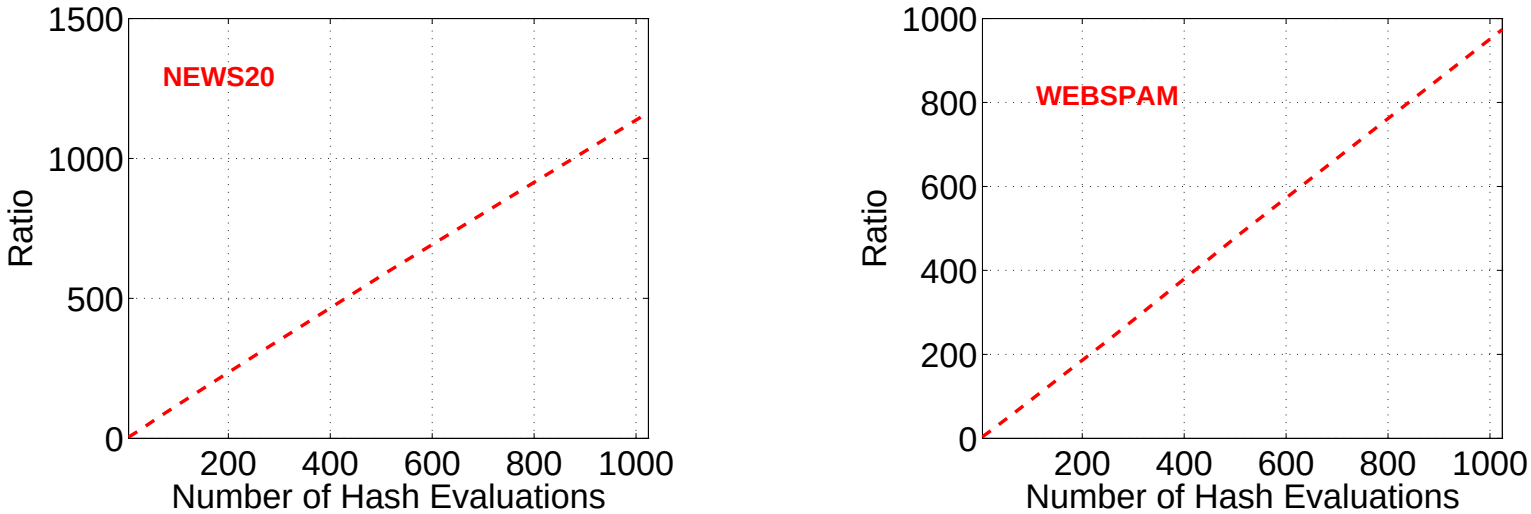


Figure: Ratio of time taken by minwise hashing to the time taken by our proposed scheme with respect to the number of hash evaluations.

The total query time with the proposed indexing scheme is $O(KL + dL)$, **an algorithmic improvement !!** over $O(dKL)$ with traditional minhash

Conclusions

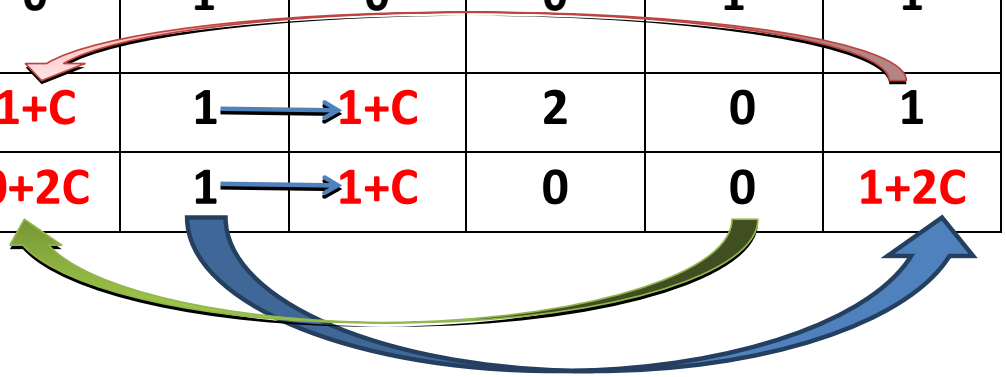
- Current large scale data processing systems deploying indexed tables based on minwise hashing suffer from the costly processing.
- We propose a new hashing scheme based on a novel "rotation" technique to densify one permutation hashes. (the idea is general).
- The obtained hash function is very similar in performance with minwise hashing, and at the same time it is an order of magnitude faster.
- As a consequence, we obtain runtime improvement over minwise hashing based near neighbor retrieval.

Adding Coin Flips Improves Variance

Remember: Implicitly re-sampling information from non-empy bins everytime we encounter simultanoue empty bins.

Insight: The sampling process is constrained because we need to ensure proper alignment. There is not enought randomness in the sampling process. Coin flips add randomness and respect proper alignments.

	Bin 0	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
Direction Bits (q)	0	1	0	0	1	1
$H^+(S_1)$	1+C	1	1+C	2	0	1
$H^+(S_2)$	0+2C	1	1+C	0	0	1+2C



Can also be done in $O(d + k)$

Improved Densification of One Permutation Hashing (Shrivastava & Li UAI 2014)

Thanks!!