Efficient Estimation for High Similarities using Odd Sketches

Michael Mitzenmacher
Harvard University
Cambridge, MA
michaelm@eecs.harvard.edu

Rasmus Pagh
IT University of Copenhagen
Copenhagen, Denmark
pagh@itu.dk

Ninh Pham
IT University of Copenhagen
Copenhagen, Denmark
ndap@itu.dk

ABSTRACT

Estimating set similarity is a central problem in many computer applications. In this paper we introduce the *Odd Sketch*, a compact *binary* sketch for estimating the Jaccard similarity of two sets. The exclusive-or of two sketches equals the sketch of the symmetric difference of the two sets. This means that Odd Sketches provide a highly space-efficient estimator for sets of high similarity, which is relevant in applications such as web duplicate detection, collaborative filtering, and association rule learning. The method extends to weighted Jaccard similarity, relevant e.g. for TF-IDF vector comparison.

We present a theoretical analysis of the quality of estimation to guarantee the reliability of Odd Sketch-based estimators. Our experiments confirm this efficiency, and demonstrate the efficiency of Odd Sketches in comparison with b-bit minwise hashing schemes on association rule learning and web duplicate detection tasks.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

Set similarity; Minwise hashing; Bloom filters

1. INTRODUCTION

Estimating set similarities is a fundamental problem in databases, machine learning, and information retrieval. Given the two sets, $\mathbb{S}_1$ and $\mathbb{S}_2$, where $\mathbb{S}_1, \mathbb{S}_2 \subseteq \Omega = \{0, 1, \ldots, D-1\}$, a challenge is how to quickly compute their Jaccard similarity coefficient J, a normalized measure of set similarity:

$$J(\mathbb{S}_1, \mathbb{S}_2) = \frac{|\mathbb{S}_1 \cap \mathbb{S}_2|}{|\mathbb{S}_1 \cup \mathbb{S}_2|}.$$

One can view large datasets of Web documents as collections of sets where sets and set elements correspond to documents and document words/shingles, respectively. Other examples are datasets encountered in recommender systems, where users and items can be viewed as sets and set elements. Hence, set similarity estimation is one of the key research challenges in many application areas, such as web duplicate detection [4, 6, 12, 18], collaborate filtering [1, 9], and association rule learning [8].

Many applications of set similarity arise in large-scale datasets. For instance, a search engine needs to crawl and index billions of web-pages. Collaborative filtering engines from sites such as Amazon or NetFlix have to deal with tens of millions of users' data. Performing similarity search over such large-scale datasets is very time-consuming. If we are willing to accept an *estimate* of J it turns out that it is possible to get by with much less computation and storage. But how much better can it get? In this paper we address the following question:

If each set $\mathbb{S}$ is summarized in a data structure $\mathbb{D}(\mathbb{S})$ of n bits, how precise an estimate of $J(\mathbb{S}_1, \mathbb{S}_2)$ is it possible to make based on $\mathbb{D}(\mathbb{S}_1)$ and $\mathbb{D}(\mathbb{S}_2)$?

Our main finding is that existing solutions, while highly efficient in general, are not optimal when J is close to 1. We present a novel solution, the *Odd Sketch*, that yields improved precision in the high similarity regime.

Although the setting where J is close to 1 has not often been the primary focus when studying similarity measures, there are many applications where this regime is important. Consider a setting where the goal is not just to find a similar item, but to provide a short list and ranking of the most similar items. For example, in the setting of document similarity, in a sufficiently rich environment there may be hundreds of documents quite similar to another document, and the user wants to see the top ten. For such applications, we require methods that are very accurate for high similarity values J.

1.1 Minwise Hashing Schemes

Because minwise hashing is a building block of our approach, and because b-bit minwise hashing is our primary alternative for comparison, we review both briefly.

1.1.1 Minwise Hashing

Minwise hashing is a powerful algorithmic technique to estimate set similarities, originally proposed by Broder et al. [4, 5]. It was used to detect and cluster similar documents in the early AltaVista search engine [6]. Since then, the scheme has been applied successfully in a variety of applications, including similarity search [4, 5, 6], association

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2568017.

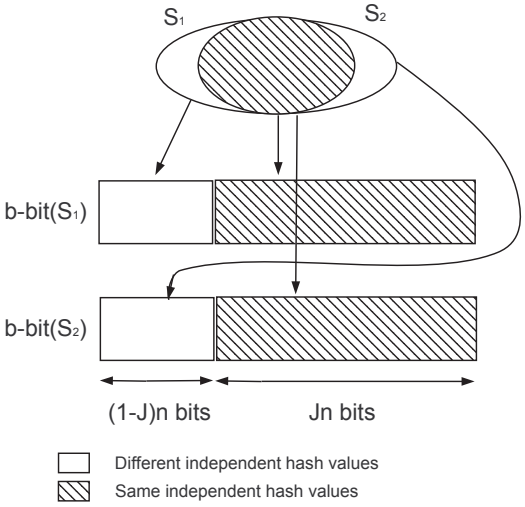

Figure 1: Illustration of the b-bit minwise hashing construction. Given a high Jaccard similarity J and two minhashes S_1, S_2, we expect that $|S_1 \cap S_2| = Jn$ (filled space) and $|S_1 \Delta S_2| = 2(1-J)n$ (white space). Due to the same independent hash values in the filled space, the error of the b-bit scheme corresponds to the error of the estimate of $|S_1 \Delta S_2|$. Inaccuracy in just a few bit positions in the white space will yield a large relative error of the estimate of J.

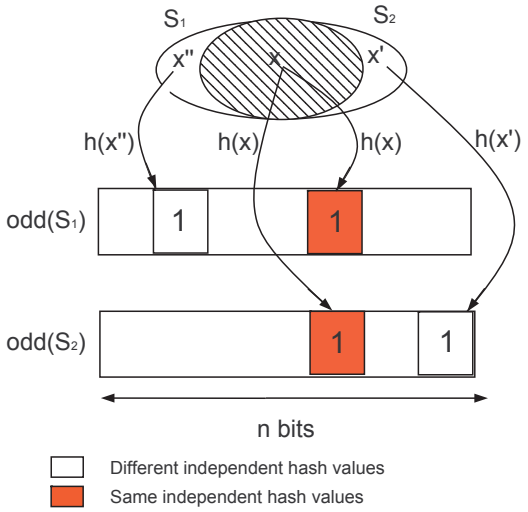

Figure 2: Illustration of the Odd Sketch construction. Odd Sketch starts with a 0s bit-vector of size n. We flip a *bit* according to each element of the minhashes S_1 and S_2. The contributions of elements in $S_1 \cap S_2$ cancel out in the exclusive-or $odd(S_1) \oplus odd(S_2)$, so Odd Sketches use all of the n bits to estimate the symmetric difference size $|S_1 \Delta S_2|$. This reduces the variance when J is close to 1.

rule learning [8], compressing social networks [7], advertising diversification [11], tracking Web spam [21], web duplicate detection [15], large-scale learning [16], and more [1, 3, 14].

We now briefly review Broder's minwise hashing scheme. Given a random permutation $\pi : \Omega \mapsto \Omega$, the Jaccard similarity of $\mathbb{S}_1$ and $\mathbb{S}_2$ is

$$J(\mathbb{S}_1, \mathbb{S}_2) = \mathbf{Pr}[\min(\pi(\mathbb{S}_1)) = \min(\pi(\mathbb{S}_2))].$$

Therefore we get an estimator for J by considering a sequence of permutations $\pi_1, \ldots, \pi_k$ and storing the annotated minimum values (called "minhashes").

$$S_1 = \{(i, \min(\pi_i(\mathbb{S}_1))) \mid i = 1, \ldots, k\},$$
$$S_2 = \{(i, \min(\pi_i(\mathbb{S}_2))) \mid i = 1, \ldots, k\}.$$

We estimate J by the fraction $\hat{J} = |S_1 \cap S_2|/k$. This estimator is unbiased, and by independence of the permutations it can be shown that $\mathbf{Var}[\hat{J}] = \frac{J(1-J)}{k}$.

Observe that a minhash can be stored as an array of length k containing the minimum for each $i = 1, \ldots, k$. The hash value $\min(\pi(\mathbb{S}))$ in the minhash is stored as an integer of typically 32 or 64 bits. That means that Broder's scheme might use $32k$ or $64k$ bits of memory to store k hash values for any set $\mathbb{S}$.

1.1.2 b-bit Minwise Hashing

At WWW'10 Li and König [15] proposed b-bit minwise hashing as a space-efficient variant of Broder's minwise hashing scheme. Instead of storing $b = 32$ or $b = 64$ bits for each permutation, this approach suggested using the lowest b bits. It is based on the intuition that the same hash values give the same lowest b bits whereas the different hash values give different lowest b bits with probability $1 - 1/2^b$. Figure 1 shows how to construct b-bit minwise sketches.

Let $\min_b(\pi(\mathbb{S}))$ denote the lowest b bits of the hash value $\min(\pi(\mathbb{S}))$. Then the b-bit minhash S_1^b is obtained from the standard minhash S_1 by replacing min by $\min_b$, reducing space usage to kb. An unbiased estimator $\hat{J}^b$ for $J(\mathbb{S}_1, \mathbb{S}_2)$ and its variance can be computed as follows:

$$\hat{J}^b = \frac{|S_1^b \cap S_2^b|/k - 1/2^b}{1 - 1/2^b} \quad , \quad \mathbf{Var}[\hat{J}^b] = \frac{1-J}{k}\left(J + \frac{1}{2^b - 1}\right).$$

However, when the Jaccard similarity is high it seems that the b-bit scheme offers less information than it might be able to. As an extreme example, suppose that we get the estimate $\hat{J}^b = 1$, i.e., all the bits of the two summaries are identical. How confident can we be that J is indeed close to 1? For example, even if we actually have $J = 1 - \frac{2}{k}$ it is quite likely that the two summaries will be identical. Somehow, since the two summaries are so highly correlated, differences of just a few bit positions will lead to very different conclusions on how close J is to 1. Thus we might ask: Is it possible to do better, avoiding the limits on accuracy that comes when summaries are highly correlated?

1.2 Our contribution

In this paper, we introduce the *Odd Sketch*, a compact binary sketch for estimating the Jaccard similarity of two sets. This binary sketch is similar to a Bloom filter with one hash function, constructed on the original minhashes with the "odd" feature that the usual disjunction is replaced by an exclusive-or operation. That is, we hash each element of the minhash into a bit-array data structure. (We will refer to the hash function used for this as the "sketch hash function".)

Now, instead of setting a bit to 1, we *flip* a bit according to the sketch hash value of each element in the minhash. We apply the Odd Sketches to minhashes, which means that the Odd Sketch records for each hash value whether it is mapped to by an odd number of elements in the minhash. Figure 2 shows a high level illustration of the construction of Odd Sketches.

A key feature of the Odd Sketch is that when we compute the exclusive-or of two Odd Sketches, the result will be equal to the Odd Sketch of the symmetric difference of the minhashes, i.e., the set of elements in one minhash but not the other. This is because the contribution of all identical elements in the minhashes cancel out. In turn, this means that we are able to base Odd Sketches of size n on minhashes of size significantly above n whenever J is close to 1 and there are many identical values in the minhashes, so that the variance induced by the minhash step is reduced.

The technical difficulty is to provide a good estimator for the size of a set based on its Odd Sketch. We provide a surprisingly simple, asymptotically precise expression for the expected fraction of 1s in an Odd Sketch, and show via concentration around this expectation that the resulting estimator has good precision as long as the fraction of 1s is bounded away from 1/2.

We note that a similar approach has previously been used to estimate the number of distinct elements in a stream, where the usual disjunction was used instead of an exclusive-or operation [2]. One of our contributions is showing that tracking the *parity* of the number of items that hash to a bucket is a useful technique in the context of estimating the size of set differences (rather than the size of sets).

2. ODD SKETCHES

2.1 Construction

The Odd Sketch is a simple, linear sketch of the indicator vector of a set S. Concretely, the sketch consists of an array s of $n > 2$ bits. Let $h : \Omega \mapsto [n]$ be a hash function, which we assume here is fully random. In the Odd Sketch, which we denote by $odd(S)$, the ith bit is given by

$$s_i = \bigoplus_{x \in S} 1_{h(x)=i} \ ,$$

where $0 \leq i < n$. That is, s_i is the parity of the number of set items that hash to the ith location.

To compute the sketch, we start with the zero bit vector of size n. Then we evaluate h on each $x \in S$, and flip the bit $s_{h(x)}$ of the sketch, as shown in the pseudo-code in Algorithm 1.

Algorithm 1 Odd Sketch(S, n)

Require: A set $S \subset \Omega$ and the size of sketch in bits n
1: $s \leftarrow [0]^n$
2: Pick a random hash function $h : \Omega \mapsto [n]$
3: **for** each set element $x \in S$ **do**
4: $\quad s_{h(x)} = s_{h(x)} \oplus 1$
5: **end for**
6: **return** s

Because $odd(S)$ records the parity of the number of elements that hash to a location, it follows that the Odd Sketch of the symmetric set difference $S_1 \Delta S_2$ is the exclusive-or of the Odd Sketches $odd(S_1)$ and $odd(S_2)$.

LEMMA 1. $odd(S_1) \oplus odd(S_2) = odd(S_1 \Delta S_2)$.

In the following section, we show how to estimate the size of a set from the number of 1s in its Odd Sketch. By Lemma 1 we can use this to estimate $|S_1 \Delta S_2|$ from the Odd Sketches of S_1 and S_2. If sets S_1 and S_2 are minhashes of $\mathbb{S}_1$ and $\mathbb{S}_2$, then we can estimate the Jaccard similarity of original two sets from the Odd Sketches of S_1 and S_2.

2.2 Estimation

2.2.1 Estimating a set's size from its Odd Sketch

Let m and n denote the size of set S and the size of $odd(S)$ in bits, respectively. Because our hash functions are fully random, we can think about the process of constructing $odd(S)$ as that of independently throwing m balls into n bins, and storing as s_i the parity of the number of balls in bin i. We are interested in generating an estimate $\hat{m}$ for m based on the observation of the number of odd bins in $odd(S)$. In the following we present two estimation approaches for the estimate $\hat{m}$. The first one is based on the Markov chain model and the second one relies on the standard *Poisson approximation* to the balls and bins setting. Both approaches yield the same estimate when n is sufficiently large.

Consider the parity of number of balls landing in any specific bin (say the first) as a simple two-state Markov chain model. The first/second state corresponds to the even/odd parity. The probability of changing states is $1/n$. Let p_i be the probability that any specific bin has an odd number of balls after i balls have been thrown. A simple induction based on Markov chains yields

$$p_i = \frac{1 - (1 - 2/n)^i}{2}.$$

It helps to now introduce some notation. Let X_i be a 0-1 random variable corresponding to the parity of the number of balls that land in the ith bin after throwing m balls, and let $X = \sum_i X_i$. We have shown that

$$\mathbf{E}[X] = n\frac{1 - (1 - 2/n)^m}{2}.$$

Hence, a seemingly reasonable approximation for m if we see z odd bins in the sketch is to assume that $z \approx \mathbf{E}[X]$, in which case

$$z \approx n\frac{1 - (1 - 2/n)^m}{2},$$

and solving we obtain an estimate $\hat{m}$ by

$$\hat{m} = \frac{\ln(1 - 2z/n)}{\ln(1 - 2/n)}. \quad (1)$$

This approximation is reasonable if X is sharply concentrated around its expectation, which we show later.

The second estimation approach leverages the standard *Poisson approximation* to the balls and bins setting and provides a pratical estimate. That is, when m balls are thrown into n bins, this is very approximately the same as independently giving each bin a number of balls that is Poisson distributed with mean $\mu = m/n$. (We discuss this further below; also, see [19, Section 5.4].) Lemma 2 provides the relationship between the Poisson distribution with mean μ and the parity of the distribution.

LEMMA 2. *(Schuster and Philippou [20]) Let Q be a random variable that has Poisson distribution with mean μ. The probability p that Q is odd is $(1 - e^{-2\mu})/2$.*

PROOF. The probability that Q is odd is given by

$$p = \sum_{i \text{ odd}} \frac{e^{-\mu}\mu^i}{i!} = e^{-\mu} \sum_{i \text{ odd}} \frac{\mu^i}{i!}$$
$$= e^{-\mu}\frac{e^{\mu} - e^{-\mu}}{2} = \frac{1 - e^{-2\mu}}{2} \quad . \quad \square$$

Let Y_i be the parity of the number of balls that land in the ith bin in the setting where the number of balls are independently Poisson distributed with mean $\mu = m/n$ in each bin, and let $Y = \sum_i Y_i$ be the number of bins with an odd number of balls. Then

$$\mathbf{E}[Y] = np = n\frac{1 - e^{-2m/n}}{2}.$$

Hence, a seemingly reasonable approximation for m if we see z odd bins in the sketch is to assume that $z \approx \mathbf{E}[Y]$, in which case we obtain an estimate $\hat{m}$ as

$$\hat{m} = -\frac{n}{2}\ln(1 - 2z/n). \tag{2}$$

Since the Y_i are independent and identically distributed, standard Chernoff bounds provide that $z \approx \mathbf{E}[Y]$ with high probability, as we clarify further below. We note that when n is sufficiently large in practice, we have $\ln(1 - 2/n) \approx -2/n$. In this case, the estimate is approximately the estimate derived from equation 1.

2.2.2 Estimating Jaccard similarity from Odd Sketches

Suppose we construct Odd Sketches $odd(S_1), odd(S_2)$ from the minhashes S_1 and S_2 derived from $\mathbb{S}_1$ and $\mathbb{S}_2$. Recall that, when we construct sets S_1 and S_2, if we think of the sets as random variables before instantiating them, we have

$$\mathbf{E}[|S_1 \Delta S_2|] = 2k(1 - J),$$

where k is the number of independent permutations and J is the Jaccard similarity of $\mathbb{S}_1$ and $\mathbb{S}_2$. Moreover, $|S_1 \Delta S_2|$ should be closely concentrated around its expectation, since each permutation independently gives a match with probability J. Once we have instantiated S_1 and S_2, given $odd(S_1)$ and $odd(S_2)$, we can estimate $|S_1 \Delta S_2|$ for the S_1 and S_2 we derived, using equation 2. For notational convenience we will think of $odd(S_1)$ and $odd(S_2)$ as the sets of bit positions containing 1, which means that their exclusive-or corresponds exactly to the symmetric difference. If we use $|S_1 \hat{\Delta} S_2|$ to denote our estimate of $|S_1 \Delta S_2|$, then

$$|S_1 \hat{\Delta} S_2| = -\frac{n}{2}\ln(1 - 2\,|odd(S_1)\Delta odd(S_2)|/n).$$

Here $|odd(S_1)\Delta odd(S_2)|$ refers to the number of 1s in the structure. Using $|S_1 \hat{\Delta} S_2|$ as a proxy for $\mathbf{E}[|S_1 \Delta S_2|]$, the Jaccard similarity can be estimated as follows:

$$\hat{J}^{odd} = 1 - \frac{|S_1 \hat{\Delta} S_2|}{2k}$$
$$= 1 + \frac{n}{4k}\ln\left(1 - \frac{2\,|odd(S_1)\Delta odd(S_2)|}{n}\right).$$

Both Odd Sketches and b-bit minwise hashing can be viewed as variations of the original minwise hashing scheme that reduce the number of bits used. The quality of their estimators is dependent on the quality of the original minwise estimators. In practice, both Odd Sketches and b-bit minwise hashing need to use more permutations but less storage space than the original minwise hashing scheme.

2.3 Analysis

In the previous section, we assumed that the number of odd bins in our data structure was closely concentrated around its expectation to justify various approximations. Here we justify this assumption. This is straightforward in the Poisson setting where bin parities are independent; we show how to handle the dependencies that exist in the balls-and-bins model. We also directly calculate the variance of the number of odd bins for both the Poisson and balls-and-bins setting.

2.3.1 Concentration

Recall our notation: we throw $m = \mu n$ balls into n bins, so that the average number of balls per bin is μ. We let X_i be the parity of the number of balls that land in the ith bin and $X = \sum_i X_i$ be the number of odd bins. Similarly, let Y_i be the parity of the number of balls that land in the ith bin in the setting where the number of balls are independently Poisson-distributed, and let $Y = \sum_i Y_i$. We show that X and Y are closely concentrated around their means.

We use the standard approach of passing to the setting where each bin obtains independently a Poisson distributed number of balls with mean μ. This is justified by, for example, [19, Corollary 5.9] where the following is shown:

LEMMA 3. *[Corollary 5.9 of [19]] Any event that takes placed with probability p when each bin obtains an independently distributed Poisson number of balls with mean μ takes place with probability at most $pe\sqrt{m}$ when $m = \mu n$ balls are thrown into n bins.*

Since Y is the sum of independent 0-1 random variables, applying a Chernoff bound [19, Exercise 4.13] to Y yields

$$\mathbf{Pr}(|Y - E[Y]| \geq \epsilon n) \leq 2e^{-2n\epsilon^2}.$$

Hence, from Lemma 3 we have

$$\mathbf{Pr}(|X - E[Y]| \geq \epsilon n) \leq (2e\sqrt{m})e^{-2n\epsilon^2}.$$

Denote by $\bar{X} = \frac{1}{n}X$ the fraction of bins with an odd number of balls. We find

$$\mathbf{Pr}(|\bar{X} - \frac{1}{n}E[Y]| \geq \epsilon) \leq (2e\sqrt{m})e^{-2n\epsilon^2} \quad ,$$
$$\mathbf{Pr}(|\bar{X} - p| \geq \epsilon) \leq (2e\sqrt{m})e^{-2n\epsilon^2},$$

where $p = \left(1 - e^{-2m/n}\right)/2$. The true expected fraction of odd bins is $\mathbf{E}[X]/n = \frac{1-(1-2/n)^m}{2}$, which differs from p by an $o(1)$ amount.

Since m corresponds to the symmetric difference between two minhashes, we have $m \leq 2k$. Hence, by choosing $n > c\epsilon^{-2}\log k$ for some constant c, our estimator closely concentrates around its mean with probability $1 - k^{-\omega(1)}$.

2.3.2 Variance bound

We note that the variance on the number of odd bins for the Poisson setting is trivial to calculate, since the bins are

independent. Letting $p = \left(1 - e^{-2m/n}\right)/2$, the standard result (on variance of biased coin flips) gives that the variance is $np(1-p)$.

For the balls and bins case there are dependencies among the bin loads that make the variance calculation more difficult. Recall that p_i is the probability that any specific bin has an odd number of balls after i balls have been thrown, and

$$p_i = \frac{1-(1-2/n)^i}{2},$$

so each $X_i = 1$ with probability $(1-(1-2/n)^m)/2$. To calculate the variance, we first calculate $\mathbf{E}[X^2]$; the standard expansion gives

$$\begin{aligned}\mathbf{E}[X^2] &= \mathbf{E}[(\sum_i X_i)^2] \\ &= \sum_i \mathbf{E}[X_i^2] + 2\sum_{i<j} \mathbf{E}[X_i X_j] \\ &= \sum_i \mathbf{E}[X_i] + 2\sum_{i<j} \mathbf{E}[X_i X_j].\end{aligned}$$

where we have used the fact that $(X_i)^2 = X_i$ since X_i only takes on the values 0-1. The first summation is just $\mathbf{E}[X]$.

To calculate the second summation, by symmetry it suffices to consider a specific pair of variables, say X_1 and X_2. We consider the total number of balls that land in the combination of bins 1 and 2. If this number is odd, then clearly $X_1X_2 = 0$. If this number is 0, then clearly $X_1X_2 = 0$. If this number is even, but more than 0, then $X_1X_2 = 1$ with probability exactly 1/2. To see this, consider the last ball that lands in either bin 1 or bin 2. One of these bins must have an odd number of balls. If the new ball lands in the other bin, then both have an odd number of balls; this happens with probability 1/2. It follows that $E[X_1X_2]$ is half the probability that bins 1 and 2 considered together obtain an even and positive number of balls. As with the calculation for p_i, a simple induction based on the two-state Markov chain model shows that after i balls have been thrown, the probability q_i that the first two bins have an even number of balls greater than 0 is

$$q_i = \frac{1+(1-4/n)^i - 2(1-2/n)^i}{2}.$$

Hence the second sum is

$$\binom{n}{2}\frac{1+(1-4/n)^m - 2(1-2/n)^m}{2}.$$

The variance is then $E[X^2] - E[X]^2$, or

$$\binom{n}{2}\frac{1+(1-4/n)^m - 2(1-2/n)^m}{2} + \frac{n(1-(1-2/n)^m)}{2} - \left(\frac{n(1-(1-2/n)^m)}{2}\right)^2.$$

Simplifying, this is

$$n^2\frac{(1-4/n)^m - (1-2/n)^{2m}}{4} + n\frac{1-(1-4/n)^m}{4}.$$

While this is easily seen to be $O(n^2)$, the coefficient

$$\frac{(1-4/n)^m - (1-2/n)^{2m}}{4}$$

of the n^2 term above is in fact $O(1/n^2)$ when $m = \mu n$. (Note that both expressions in the numerator converge to and are approximately $e^{-4m/n}$. By examining the asymptotics carefully one can show the coefficient is $O(1/n^2)$.) Hence the variance here is also $O(n)$. Indeed, the second term is

$$n\frac{1-(1-4/n)^m}{4} \approx n\frac{1-e^{-4m/n}}{4} = np(1-p),$$

which is the variance for the Poisson setting, and the first term is negative. Again, by considering the asymptotic expansions carefully one obtains that the variance in the Poisson case is larger than in the case where there are exactly m balls thrown for large enough n, as one might naturally expect.

2.4 Accuracy of the estimator

In the previous sections we bounded the variance and gave strong tail bounds for the fraction z/n of 1s in an Odd Sketch. Recall that its expected value is $p_m = \frac{1-(1-2/n)^m}{2}$ derived from the Markov chain and its practical estimate is $p = \frac{1-e^{-2m/n}}{2}$ derived from the Poisson approximation. What remains is to bound the error resulting from applying the estimator from equation (2), repeated here for convenience:

$$\hat{m} = -\frac{n}{2}\ln\left(1-2z/n\right).$$

Defining the function $f(x) = -\frac{n}{2}\ln(1-2x)$, we have $\hat{m} = f(z/n)$. There are two sources of inaccuracy: The first is that the estimator has a bias since the expected number of 1s, $np_m = n\frac{1-(1-2/n)^m}{2}$, differs from the practical estimate $np = n\frac{1-e^{-2m/n}}{2}$. However, it can be confirmed by an easy computation that the difference can be at most 1, so this is not significant.

The second, and more significant, source of error is that when z/n deviates from its expectation p_m, $f(z/n)$ will deviate from $f(p_m)$. Informally, an if z/n deviates from its expectation by ε, this will give an error of roughly $f'(z/n)\cdot\varepsilon$, as long as ε is small enough, where $f'(x) = \frac{n}{1-2x}$ is the derivative of f. It is clear that a small error can be magnified significantly if z/n is close to 1/2, since $f'(x)$ goes to infinity as $x \to 1/2$. Therefore we choose parameters such that p, the practical estimate of z/n, is bounded away from 1/2 ($p \approx 0.3$ gives the best results, as we see when we discuss our experiments). By the results in section 2.3.1 this means that with high probability (wrt. n) we will have $z/n < 0.4$ (say). As long as this is the case, since f' is monotonely increasing, we have that the error is bounded by $f'(0.4)|z/n - p_m| = 5n$. This bound is pessimistic, but shows that the estimation error is (with high probability) proportional to the error in the estimate of p_m. In turn, this implies that the variance of the estimator is $O(n)$.

2.5 Weighted similarity

Odd Sketches work with any notion of similarity that can be transformed to Hamming distance of two vectors. In particular, it works with any similarity measure that can be captured using the probability that two minhashes are identical. For example, the Jaccard similarity of two vectors $v, w \in \mathbf{R}^d$ with nonnegative entries can be defined as:

$$J(v,w) = \frac{\sum_i \min(v_i, w_i)}{\sum_i \max(v_i, w_i)},$$

generalizing standard Jaccard similarity which corresponds to 0-1 vectors. Hash functions that result in minhash equality with probability $J(v, w)$ can be found in [10, 13, 17].

3. EXPERIMENTAL RESULTS

We implemented b-bit minwise hashing and Odd Sketch in Matlab, and conducted experiments on a 2.67 GHz Core i7 Windows machine with 3GB of RAM. We compared the performance of b-bit minwise hashing and Odd Sketch on association rule learning and web duplication detection tasks. All results are the averages of 10 runs of the algorithms.

3.1 Parameter setting

It is obvious that the performance of both b-bit minwise and Odd Sketch depends on the number of independent permutations used in the original minwise hashing scheme. The b-bit minwise scheme uses $k_b = n/b$ permutations where the storage space is n bits and $b \geq 1$ is the number of bits per permutation. Since larger k_b provides higher accuracy, setting $b = 1$ turns out to achieve the smallest variance, as will be clear from our empirical evaluation.

In the Odd Sketch setting, the number of independent permutations k_{odd} is dependent on the sketch size n and the user-defined similarity threshold J_0. Typically, we are interested in retrieving pairs of sets such that $J > J_0$ (and perhaps subject these pairs to additional filtering). Moreover, we want to choose k_{odd} as large as possible to reduce the error from the original minwise hashing step. It seems difficult to mathematically establish the optimal way of choosing k_{odd}, but we conducted experiments that indicate that the smallest variance is achieved when the exclusive-or of two odd sketches with similarity J_0 contains around 30% 1s.

Our estimator needs the fraction of 1s in $odd(S_1 \Delta S_2)$ to be smaller than 1/2. If a fraction of more than 1/2 is observed this is (with high probability) a sign of very low Jaccard similarity, so we may estimate $J = 0$. Recall that the process of constructing $odd(S_1 \Delta S_2)$ corresponds to throwing $|S_1 \Delta S_2|$ balls into n bins. It turns out that if we choose k_{odd} such that $|S_1 \Delta S_2| \approx n/2$ we get the most accurate estimate when similarity is around J. In other words, we choose the parameter k_{odd} such that the ratio $\alpha = \frac{2k_{odd}(1-J_0)}{n} \approx \frac{1}{2}$.

We conducted experiments to evaluate this choice of ratio α. We compared the mean square error (MSE, incorporating both variance and bias) of our estimator $\hat{J}^{odd}$ for different ratios of $\alpha = \frac{2k_{odd}(1-J_0)}{n}$ in $[0.25, 1]$, and for different sketch sizes $n \in [500, 1000]$ bits. For each choice we found that $\alpha = 1/2$ gave the smallest observed MSE. Figure 3 displays the average MSE of $\hat{J}^{odd}$, averaged over variety of values of J and n. It illustrates that Odd Sketch achieves the highest accuracy when using the ratio $\alpha = 0.5$. So we can choose $k_{odd} = \frac{n}{4(1-J_0)}$ given a threshold J_0. When we are interested in $J_0 \geq 0.75$, we can set $k_{odd} > n$. This means that Odd Sketch can use more independent permutations than b-bit schemes. In fact, even for the inferior choice of $k_{odd} = n$, Odd Sketch can achieve better performance than 1-bit minwise hashing when $J_0 > 0.75$.

3.2 Accuracy of Estimation

This subsection shows experiments to further evaluate the accuracy of our estimation algorithm. We carried out experiments to compare the accuracy of b-bit minwise hashing and Odd Sketch. In the b-bit schemes, we set $k_b = n/b$ to achieve a space usage of n bits. For the Odd Sketch, we set $k_{odd} = \frac{n}{4(1-J)}$. We again measured the mean square error (MSE) of estimators of both approaches. We varied n in $\{512, 1024\}$ bits and conducted experiments on synthetic datasets. (But note that since we apply hashing the outcome is independent of the particular set elements, and we expect the same result on any real-life dataset.) This dataset is very high-dimensional ($D = 10,000$) and highly sparse (sparsity $> 99.9\%$).

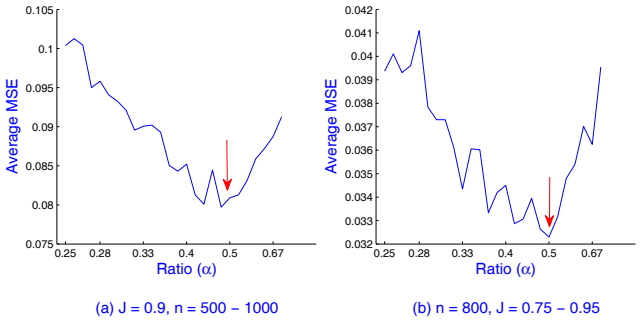

Figure 3: Comparison of the average MSE on different ratios α on the synthetic dataset: (a) Fix $J = 0.9$ and change $n = 500 - 1000$ bits; (b) Fix $n = 800$ bits and change $J = 0.75 - 0.95$.

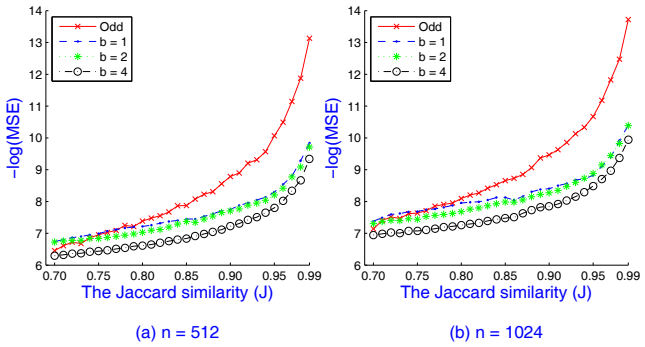

Figure 4: Comparison of the negative log of mean square error (MSE) of Odd Sketch and b-bit minwise hashing for different Jaccard similarity. In these experiments Odd Sketch used $k_{odd} = \frac{n}{4(1-J)}$ permutations, and b-bit minwise hashing used $k_b = \frac{n}{b}$.

Figure 4 shows the negative log of MSE ($-\log(\text{MSE})$) of estimators of the two approaches for different values J. We note that the MSE is always smaller than 1 in our experiments, so larger $-\log(\text{MSE})$ is better. For high Jaccard similarities $J \geq 0.8$, Odd Sketch provides a smaller error than the b-bit minwise approach. The difference is more dramatic when J is very high because Odd Sketch makes use of a larger number of independent permutations than the b-bit minwise schemes. This figure also shows that the 1-bit scheme has superior performance compared to the b-bit schemes for $b > 1$. We note that b-bit schemes for $b > 1$ require additional bit-manipulation to pack b bits of hash values into 64-bit (or 32-bit) words. In contrast, both

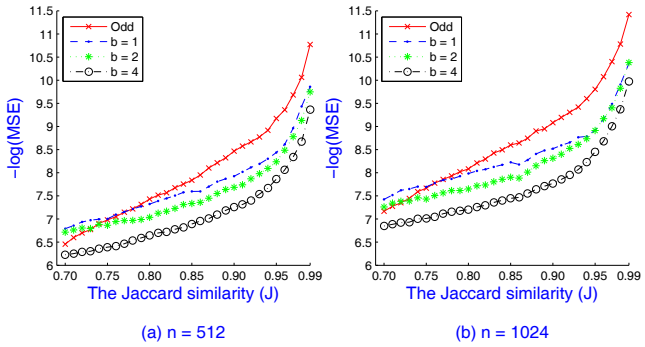

Figure 5: Comparison of the negative log of mean square error (MSE) of Odd Sketch and b-bit minwise hashing for different Jaccard similarities. Here, Odd Sketch uses $k_{odd} = n$, and b-bit minwise hashing uses $k_b = \frac{n}{b}$ permutations.

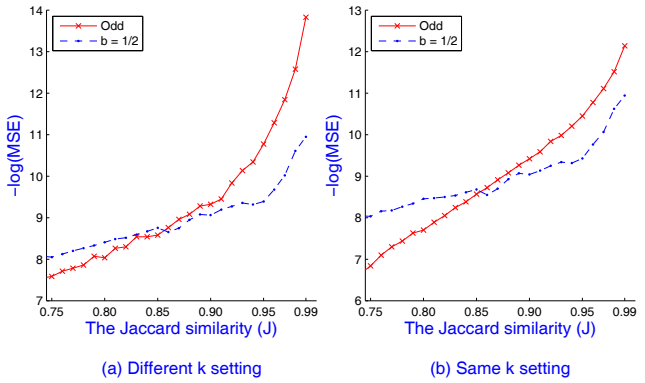

Figure 6: Comparison of the negative log of mean square error (MSE) of Odd Sketch and $\frac{1}{2}$-bit minwise hashing for different Jaccard similarities: (a) Different number of permutations: $k_{odd} = \frac{n}{4(1-J)}$ and $k_{\frac{1}{2}} = 2n$; (b) Same number of permutations: $k_{odd} = k_{\frac{1}{2}} = 2n$.

Odd Sketch and 1-bit schemes only need the XOR and bit-counting operations to compare two summaries.

One might argue that Odd Sketch requires a more expensive preprocessing step than b-bit minwise hashing due to the use of larger number of permutations in the minwise hashing step. But even with $k_{odd} = n$, where the hashing cost is identical to that of 1-bit minwise hashing, Odd Sketch still provides better accuracy when $J > 0.75$, as shown in Figure 5.

When the target similarity is very high, the authors of b-bit minwise hashing also discussed the idea of combining any 2 bits of a 1-bit minhash by XOR operations to increase the amount of information in each bit. This approach is called $\frac{1}{2}$-bit minwise hashing, and similar to Odd Sketch has a nonlinear estimator. The $\frac{1}{2}$-bit scheme uses $k_{\frac{1}{2}} = 2n$ permutations.

We carried out experiments to compare the mean square errors of estimators of Odd Sketch and $\frac{1}{2}$-bit minwise hashing, as shown in Figure 6. The figure shows that Odd Sketch achieves a considerably smaller error than $\frac{1}{2}$-bit minwise hashing when $J > 0.85$ for both choices of k. It also shows that Odd Sketch with the best choice of k_{odd} provides higher accuracy than for $k_{odd} = 2n$.

We conclude the accuracy evaluation of Odd Sketch by depicting the empirical cumulative distribution function (cdf) of estimators. Figure 7 shows the empirical cdfs of Odd Sketch, 1-bit scheme and $\frac{1}{2}$-bit scheme on 10,000 estimators of the Jaccard similarity $J = 0.9$. The slope of cdf of Odd Sketch is steeper than that of $\frac{1}{2}$-bit scheme and significantly steeper than that of 1-bit scheme. This means that Odd Sketch provides superior performance compared to b-bit minwise hashing when the target similarity is high.

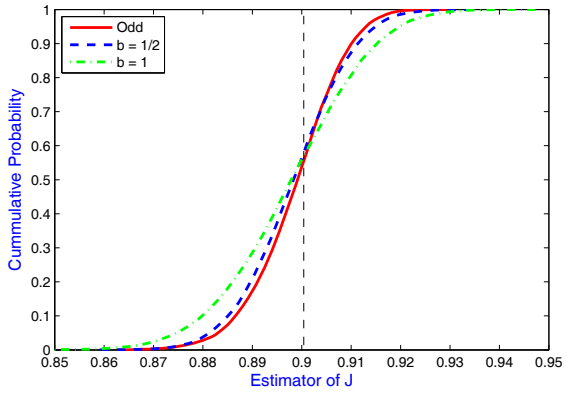

Figure 7: Comparison of the empirical cumulative distribution function (cdf) of estimators based on Odd Sketch, 1-bit scheme, and $\frac{1}{2}$-bit scheme with $J = 0.9$.

3.3 Association Rule Learning

Cohen et al. [8] used minwise hashing to generate the candidate sets of high Jaccard similarity in the context of learning pairwise associations. This subsection compares the performance of Odd Sketch and b-bit schemes in this setting. Since $b = 1$ provides the highest accuracy among $b \geq 1$, we only used the 1-bit scheme in our experiment. For a more clear comparison, we used the same number of permutations for the two approaches. We measured the precision-recall ratio of both approaches on detecting the pairwise items that have Jaccard similarity larger than a threshold J_0. We conducted experiments on the large public datasets[1]: mushroom ($N = 8124; D = 119$) and connect ($N = 67,557; D = 127$). Due to the similar results on both datasets, we only report the representative results of mushroom dataset here.

Figure 8 shows the precision-recall ratio of the Odd Sketch and the 1-bit scheme. For the high target threshold $J_0 = 0.9$, the Odd Sketch provides significantly higher precision and recall ratio (up to 10% better) than 1-bit minwise hashing. For $J_0 = 0.8$, the Odd Sketch is still better in precision but slightly worse in recall.

[1]http://fimi.ua.ac.be/data/

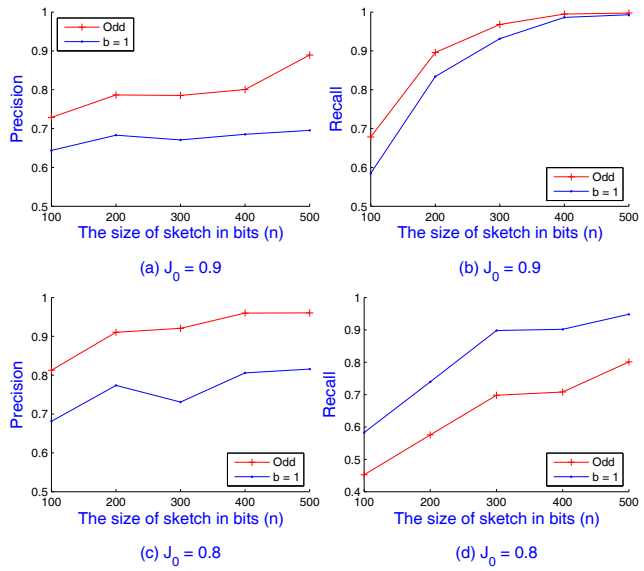

Figure 8: Comparison of the precision-recall ratio between Odd Sketches and 1-bit scheme on the mushroom dataset.

Figure 9 demonstrates the superiority of Odd Sketch compared to $\frac{1}{2}$-bit minwise hashing with respect to precision. The Odd Sketch achieved up to 20% higher precision while providing similar recall.

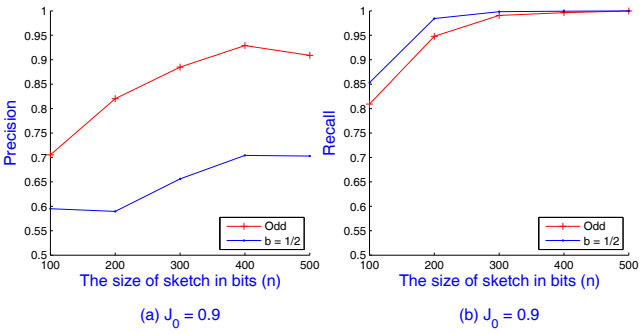

Figure 9: Comparison of the precision-recall ratio between Odd Sketches and $\frac{1}{2}$-bit scheme on the mushroom dataset with $J_0 = 0.9$.

3.4 Web Duplicate Detection

In this experiment, we compare the performance of the two approaches on web duplicate detection tasks on the bag of words dataset[2]. We picked three datasets, including KOS blog entires ($D = 6906; N = 3430$), Enron Emails ($D = 28,102; N = 39,861$), and NYTimes articles ($D = 102,660; N = 300,000$). We computed all pairwise Jaccard similarities among documents, and retrieved every pair with $J > J_0$. For the sake of comparison, we used the same number of permutations and considered the thresholds $J_0 = 0.85$ and $J_0 = 0.90$. We again used the precision-recall ratio as our standard measure.

[2]`http://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

Figures 10 and 11 show the precision-recall ratio for the Odd Sketch and 1-bit minwise hashing on three datasets with $J_0 = 0.85$ and $J_0 = 0.9$, respectively. The Odd Sketch obtains higher relative precision ratio or at least is comparable to 1-bit scheme when $J_0 = 0.85$. It achieves up to 7% and 1% higher than the 1-bit scheme on KOS blog entries and Enron Emails, respectively. For $J_0 = 0.9$, the precision ratios are almost the same on three datasets. However, Odd Sketch greatly outperforms in the recall ratio. The Odd Sketch's relative recall is approximately 15% higher than the 1-bit scheme on the KOS blog entries when $J_0 = 0.85$ and $J_0 = 0.9$. The difference in relative recall is not significant on the other datasets. These relative gaps are around 5% and less 1% on Enron Emails and NYTimes articles, respectively.

Figure 12 shows the observed precision-recall graphs of the Odd Sketch and the $\frac{1}{2}$-bit scheme. We again set $k_{odd} = k_1 = 2n$ for the sake of fair comparison. Both approaches achieve very high precision (higher than 90%) on the three datasets. The Odd Sketch still obtains higher precision than the $\frac{1}{2}$-bit scheme although the difference is not dramatic. The gap of both schemes in the recall ratio is considerable on KOS blog entries and Enron Emails. The most dramatic difference is around 4% when $n = 200$. On the NYTimes articles dataset, the ratio curves of both schemes are overlapping when $n \geq 200$.

4. CONCLUSION

In this paper, we proposed the *Odd Sketch*, a compact binary sketch for estimating the Jaccard similarity of two sets. By combining the minwise hashing technique with a hash table where only the parity of the number of items hashed to bucket is stored, Odd Sketches can be combined with just an exclusive-or operation to allow a simple estimation of the Jaccard similarity that provides a highly space-efficient solution, particularly for the high similarity regime. We presented a theoretical analysis of the quality of estimate. Our experiments on synthetic and real world datasets demonstrate the efficiency of Odd Sketches in comparison with b-bit minwise hashing schemes on association rule learning and web duplicate detection tasks. We expect that there are many other additional applications where Odd Sketches can be similarly applied.

5. ACKNOWLEDGMENTS

The work of the first author is supported by NSF grants CCF-1320231, CNS-1228598, IIS-0964473, and CCF-0915922. Part of the work by the first author was done while visiting the IT University of Copenhagen. The work of the second and third author is supported by the Danish National Research Foundation under the Sapere Aude program. We thank the anonymous reviewers for their constructive comments and suggestions.

6. REFERENCES

[1] Y. Bachrach, E. Porat, and J. S. Rosenschein. Sketching techniques for collaborative filtering. In *IJCAI*, pages 2016–2021, 2009.

[2] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*, pages 1–10. Springer, 2002.

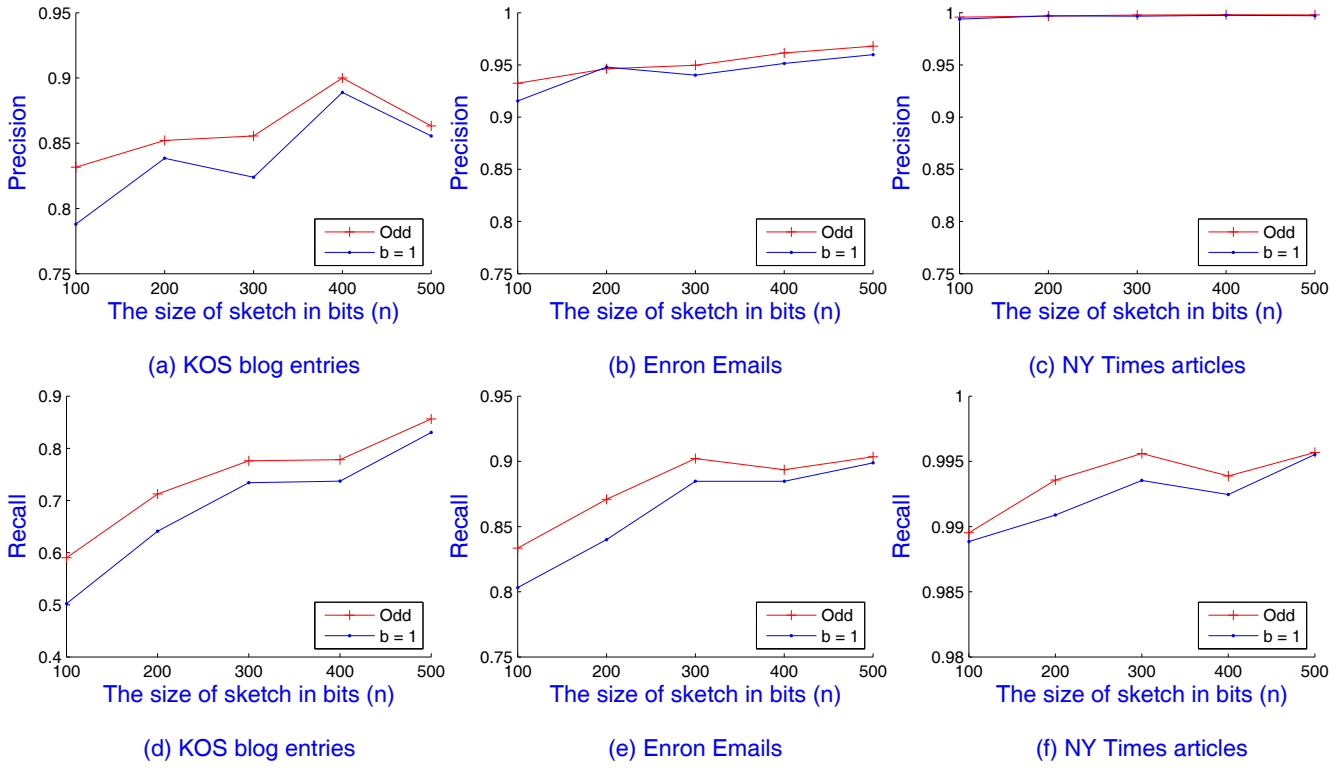

Figure 10: Comparison of the precision-recall ratio between Odd Sketches and 1-bit minwise hashing with $J_0 = 0.85$ on the three datasets: KOS blog entries, Enron Emails and NYTimes articles.

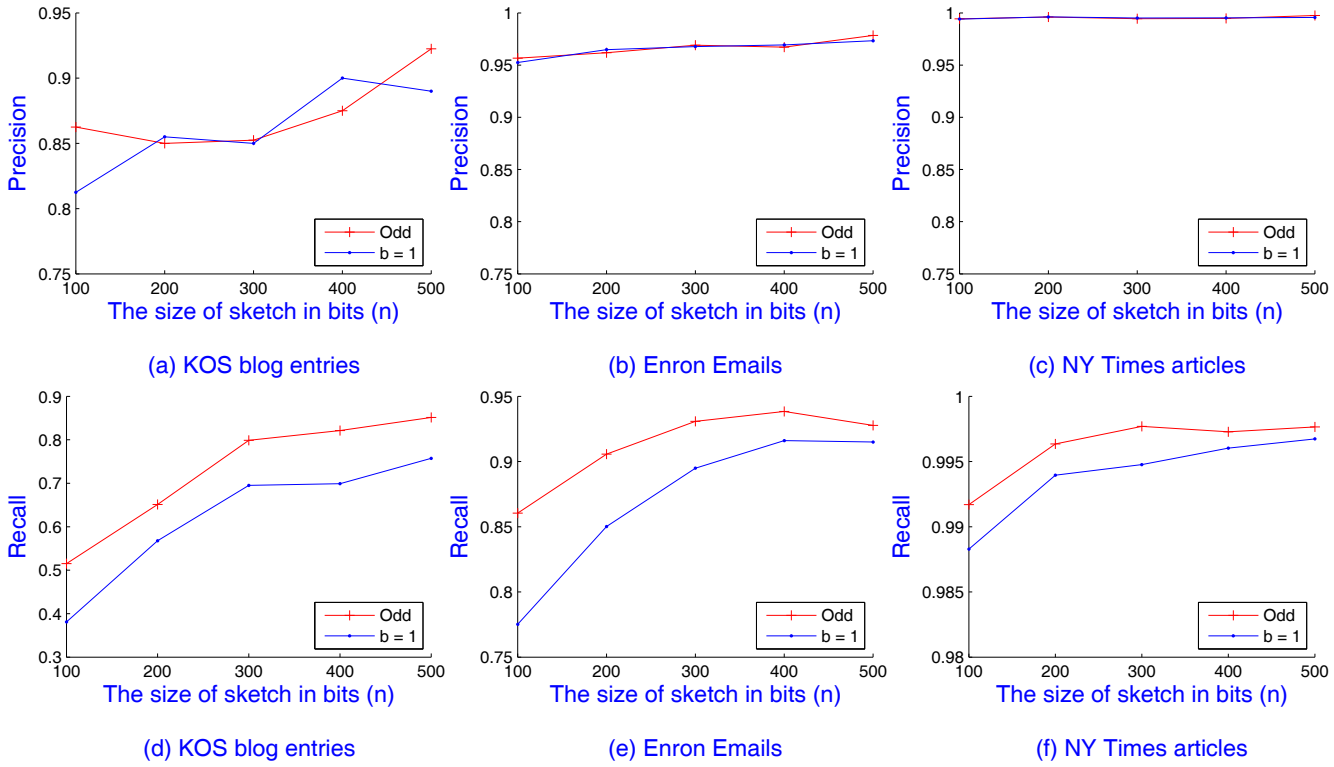

Figure 11: Comparison of the precision-recall ratio between Odd Sketches and 1-bit minwise hashing with $J_0 = 0.9$ on the three datasets: KOS blog entries, Enron Emails and NYTimes articles.

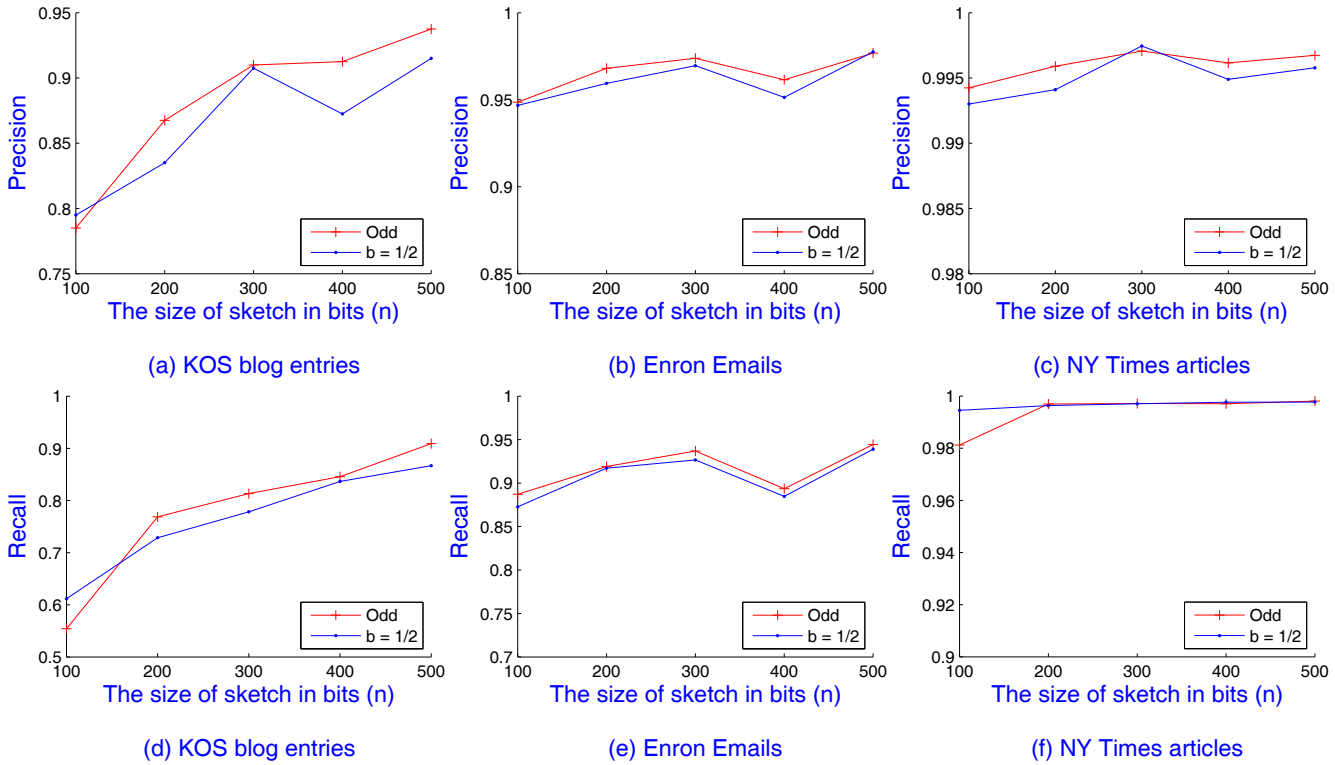

Figure 12: Comparison of the precision-recall ratio between Odd Sketches and $\frac{1}{2}$-bit minwise hashing with $J_0 = 0.9$ on the three datasets: KOS blog entries, Enron Emails and NYTimes articles.

[3] M. Bendersky and W. B. Croft. Finding text reuse on the web. In *WSDM*, pages 262–271, 2009.

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

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *J. Comput. Syst. Sci.*, 60(3):630–659, 2000.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

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

[8] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng.*, 13(1):64–78, 2001.

[9] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280, 2007.

[10] S. Gollapudi and R. Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *CIKM*, pages 475–482, 2006.

[11] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.

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

[13] S. Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *ICDM*, pages 246–255, 2010.

[14] K. Kutzkov, A. Bifet, F. Bonchi, and A. Gionis. Strip: stream learning of influence probabilities. In *KDD*, pages 275–283, 2013.

[15] P. Li and A. C. König. b-bit minwise hashing. In *WWW*, pages 671–680, 2010.

[16] P. Li, A. Shrivastava, J. L. Moore, and A. C. König. Hashing algorithms for large-scale learning. In *NIPS*, pages 2672–2680, 2011.

[17] M. Manasse, F. McSherry, and K. Talwar. Consistent weighted sampling. *MSR-TR-2010-73 technical report*, 2010.

[18] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In *WWW*, pages 141–150, 2007.

[19] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[20] E. F. Schuster and A. N. Philippou. The odds in some odd-even games. *The American Mathematical Monthly*, 82:646–648, 1975.

[21] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne. Tracking web spam with html style similarities. *TWEB*, 2(1), 2008.