using the Hamming distance between their binary sketch vectors $\mathbf{u}_s, \mathbf{v}_s \in \{0,1\}^d$. Due to (Charikar, 2002), we have

$$\cos(\mathbf{u}, \mathbf{v}) = \cos\left[\left(\frac{\pi}{d}\right) \mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)\right]. \tag{2}$$

3. Our contribution

Given a pair of real-valued vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{\mathtt{d}}$, Simhash compresses them into a pair of binary vectors $\mathbf{u}_s, \mathbf{v}_s \in \{0,1\}^d$ such that the Hamming distance between $\mathbf{u}_s$ and $\mathbf{v}_s$ gives an estimate of the Cosine similarity between $\mathbf{u}, \mathbf{v}$, where $d \ll \mathtt{d}$. A precise relation is mentioned in Equation 2. In our proposed algorithm *Simsketch*, we further compress the binary vectors $\mathbf{u}_s, \mathbf{v}_s$ to $\mathbf{u}^{(\mathbf{s})}, \mathbf{v}^{(\mathbf{s})} \in \{0,1\}^{d^{(s)}}$, where $d^{(s)} \ll d$, while maintaining an estimate of the Hamming distance between $\mathbf{u}_s, \mathbf{v}_s$. This as a consequence, maintains an estimate of the Cosine similarity between $\mathbf{u}$ and $\mathbf{v}$. Therefore, *Simsketch* further compresses (to a much smaller dimension) the hash codes obtained from Simhash and simultaneously maintains an estimate of the Cosine similarity between original real-valued data points. *Simsketch* independently can also be seen as dimensionality reduction for binary data which approximates the Hamming distance between the data points. This result can be potentially used for compressing the hashcodes obtained from other discrete hashing algorithms such as "Winner Takes All (WTA) (Yagnik et al., 2011)".

3.1. Simsketch

For a d-dimensional binary vector $\mathbf{u}_s \in \{0,1\}^d$, our algorithm reduces it to a $d^{(s)}$-dimensional binary vector $\mathbf{u}^{(\mathbf{s})} \in \{0,1\}^{d^{(s)}}$, where $d^{(s)}$ is specified later. It randomly maps each bit position $1 \leq i \leq d$ to an integer $1 \leq j \leq d^{(s)}$. In order to compute the j-th bit of $\mathbf{u}^{(\mathbf{s})}$, it checks which bit positions have been mapped to j, computes the parity of the bits located at those positions and assigns it to $\mathbf{u}^{(\mathbf{s})}[j]$.

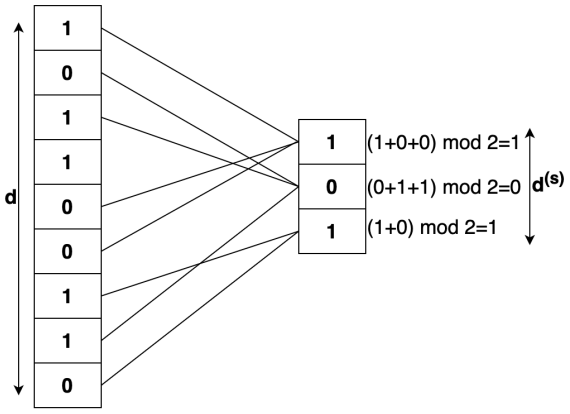

Figure 1: Illustration of Simsketch. The left binary vector is the d-dimensional binary vector obtained after applying Simhash. The right binary vector is the $d^{(s)}$ dimensional binary vector obtained after applying Simsketch on the left one.

3.2. Theoretical analysis of Simsketch

In what follows, we analyze the guarantee offered by our algorithm. This evaluation of our result is adapted from the well-studied literature of *bin-ball analysis* (Johnson and Kotz, 1977), and some steps of our analysis are similar to the result of (Mitzenmacher et al., 2014). However, we argue that the objectives of both the results are different. The algorithm proposed in (Mitzenmacher et al., 2014) takes as input the sketch obtained by applying Minhash (Broder et al., 1998) on a pair of sets, and produces as output a pair of compact binary vectors that maintains an estimate of the Jaccard Similarity between the given sets. Simsketch, however, takes as input the sketch obtained by applying Simhash on the given pair of real-valued vectors, and outputs compact binary vectors for estimating the Cosine similarity between the given original vectors. We present our analysis as follows.

Let ψ denote the number of 1s in $\mathbf{u}_s$. Using the aforementioned sketching algorithm, the probability that a particular bit position in $\mathbf{u^{(s)}}$ has value 1 is $\frac{1-(1-2/d^{(s)})^{\psi}}{2}$ (follows *via* a simple induction based on Markov chains). Let X_i be a $0-1$ random variable denoting the value of i-th bit position of $\mathbf{u^{(s)}}$, and $X = \sum_i X_i$. Then $\mathbb{E}[X] = d^{(s)}\left(\frac{1-(1-2/d^{(s)})^{\psi}}{2}\right)$. If α is a good approximation of $\mathbb{E}[X]$, that is $\alpha \approx d^{(s)}\left(\frac{1-(1-2/d^{(s)})^{\psi}}{2}\right)$, then solving this expression for ψ gives its estimate $\widetilde{\psi}$, which is

$$\widetilde{\psi} = \frac{\ln(1-2\alpha/d^{(s)})}{\ln(1-2/d^{(s)})} = -\frac{d^{(s)}}{2}\ln\left(1-\frac{2\alpha}{d^{(s)}}\right). \tag{3}$$

The second equality is satisfied when $d^{(s)}$ is sufficiently large.

Our aim is to estimate the Cosine similarity $\mathcal{S}$ between two real-valued vectors $\mathbf{u}$ and $\mathbf{v}$. For a pair of vectors $\mathbf{u}_s, \mathbf{v}_s \in \{0,1\}^d$, ψ can be considered as the Hamming distance between them. Once the vectors $\mathbf{u^{(s)}}$ and $\mathbf{v^{(s)}}$ are obtained after applying Simsketch on $\mathbf{u}_s, \mathbf{v}_s$, we can refer α as the Hamming distance between $\mathbf{u}_s, \mathbf{v}_s$. We wish to find an estimate of $\mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)$ using $\mathrm{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})$ which we estimate by considering a vector obtained by taking bitwise-XOR between $\mathbf{u}_s$ and $\mathbf{v}_s$, and putting it in Equation 3. Further, as a consequence, an estimate of $\mathcal{S}$ can be obtained using the result of Equation 2. If we denote $\widetilde{\mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)}$ as an estimate of $\mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)$, then due to Equation 3 we have

$$\widetilde{\mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)} = -\frac{d^{(s)}}{2}\ln\left(1-\frac{2\mathrm{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})}{d^{(s)}}\right). \tag{4}$$

As we have $\widetilde{\mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)}$, putting this in Equation 2, gives an estimate $\tilde{S}$ of the Cosine similarity $\mathcal{S}$ between two real-valued vectors $\mathbf{u}$ and $\mathbf{v}$.

$$\begin{aligned}\tilde{S} &= \cos\left[\left(\frac{\pi}{d}\right)\widetilde{\mathrm{Ham}(\mathbf{u}_s, \mathbf{v}_s)}\right]. \\ &= \cos\left[-\left(\frac{\pi}{2}\right)\frac{d^{(s)}}{d}\ln\left(1-\frac{2\mathrm{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})}{d^{(s)}}\right)\right].\end{aligned} \tag{5}$$

The first equality follows from Equation 2, and the second from Equation 4. Due to Equation 5, our estimate $\tilde{S}$ of the Cosine similarity is closely related to the estimate of the quantity $\mathrm{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})/d^{(s)}$. In what follows, we present the concentration and the variance bounds of this quantity.

Concentration bounds. Our sketching algorithm can be considered as an experiment of throwing some balls into several bins and the sketch is obtained by considering the parity of the number of balls that have fallen into each bin. The sketch of the input vector $\mathbf{u}_s$ is obtained by throwing ψ balls into $d^{(s)}$ bins and the sketch of each bin corresponds to the parity of the number of balls that have fallen into that particular bin. A fundamental difficulty in analyzing this scenario is that the events – number of balls falling into each bin – are not independent. It is important to overcome these sorts of dependencies. We circumvent this problem by considering the scenario where each bin receives a number (of balls) independently from a Poisson distribution having mean $\psi/d^{(s)}$. The difference between throwing ψ balls randomly and assigning each bin a number of balls from a Poisson distribution with mean $\psi/d^{(s)}$ is that in the former, we know that there are ψ balls in total whereas in the latter we only know that ψ is the expected number of balls in all of the bins. We use the following result.

Lemma 1 (Corollary 5.9 of (Mitzenmacher and Upfal, 2005)) *Any event that takes place with probability p where each bin obtains an independently distributed Poisson number of balls with mean μ, occurs with probability at most $pe\sqrt{m}$ when $m = \mu n$ balls are thrown into n bins.*

Let X_i be a random variable denoting the parity of the number of balls that fall in the i-th bin, and $X = \sum_i X_i$ be a random variable denoting the number of bins containing an odd number of balls – let us call them *odd bins*. Similarly, let Y_i be the parity of the number of balls that fall in the i-th bin in the setting where the number of balls are independently Poisson-distributed, and let $Y = \sum_i Y_i$. Due to Chernoff bounds, we have,

$$\Pr([Y - \mathbb{E}[Y]] \geq \epsilon d^{(s)}) \leq 2e^{-2d^{(s)}\epsilon^2}. \tag{6}$$

Equation 6 along with Lemma 1 gives the following

$$\Pr([X - \mathbb{E}[Y]] \geq \epsilon d^{(s)}) \leq (2e\sqrt{\psi})^{-2d^{(s)}\epsilon^2}.$$

The term $\mathbb{E}[Y]/d^{(s)}$ is the mean when the Poisson distribution is considered and its value is $1 - e^{-\frac{2\psi}{d^{(s)}}}$, while the term $X/d^{(s)}$ is the true expected fraction of the number of odd bins, and its value is $\frac{1-(1-2/d^{(s)})^{\psi}}{2}$.

We remark that the guarantee stated in Equation 4 holds for binary vectors and Hamming distance as the similarity measure, and can be used in any other applications which require compressing binary vectors preserving Hamming distance. Furthermore, the guarantee offered by Equation 4 is different from the well-known Locality Sensitive Hashing (LSH) algorithm due to Gionis *et. al.* (Gionis et al., 1999). Equation 4 suggests compressing high dimensional binary vectors into low-dimensional binary vectors such that from the low-dimensional binary vectors we can accurately estimate the corresponding pairwise Hamming distance of full-dimension. Our guarantee holds for both close and far distances (for far points the compressed dimension $d^{(s)}$ increases as per Equation 4) whereas the guarantee of LSH holds for close points only.

Second moment estimation. We would like to give a variance bound on the estimate of the number of odd bins. Recall that X_i is a $0-1$ random variable denoting the parity of the number of balls landed in the i-th bin, and $X_i = 1$ with probability $\frac{1-(1-2/d^{(s)})^{\psi}}{2}$. The random variable $X = \sum_i X_i$ denotes the number of odd bins. We wish to calculate the variance of the random variable X, which is $\mathbb{E}[X^2] - \mathbb{E}[X]^2$. We start with giving a bound on the term $\mathbb{E}[X^2]$.

$$\begin{aligned}\mathbb{E}[X^2] = \mathbb{E}[(\sum_i X_i)^2] &= \sum_i \mathbb{E}[X_i^2] + 2\sum_{i<j} \mathbb{E}[X_i X_j]. \\ &= \sum_i \mathbb{E}[X_i] + 2\sum_{i<j} \mathbb{E}[X_i X_j].\end{aligned} \tag{7}$$

The last equality follows as X_i is a $0-1$ random variable, thus $X_i^2 = X_i$, and as a result $\mathbb{E}[X_i^2] = \mathbb{E}[X_i]$. We would like to give a bound on the second term of Equation 7. Let us consider a specific pair of random variables, say X_1 and X_2, that correspond to the parities of the number of balls that have fallen into the first and second bins. We wish to evaluate the possibilities that the random variable X_1X_2 attains the value 1. Clearly, if the total number of balls that landed in the first two bins is odd, then $X_1X_2 = 0$, because at least one of the bins will have an even number of balls, which leads to the value of the corresponding random variable becoming zero, causing the value of their product to be zero. The only possible case when X_1X_2 attains the value 1 is when both the first as well as the second bin contain an odd number of balls. This happens with probability $1/2$. To understand this, consider the last ball that lands in either of the first two bins. One of these bins must have an odd number of balls. If the ball falls into the other bin, then both the bins have an odd number of balls and this happens with probability $1/2$. Thus, the probability that both the bins have an odd number of balls after the i balls have been thrown is

$$\frac{1 + (1 - 4/d^{(s)})^i - 2(1 - 2/d^{(s)})^i}{2}.$$

The above expression follows by applying a simple induction on the two-state Markov chain. The value of the second term in Equation 7 is

$$\binom{d^{(s)}}{2} \frac{1 + (1 - 4/d^{(s)})^{\psi} - 2(1 - 2/d^{(s)})^{\psi}}{2}.$$

Thus, the variance of our estimate

$$\begin{aligned}&\mathbb{E}[X^2] - \mathbb{E}[X]^2 = \sum_i \mathbb{E}[X_i] + 2\sum_{i<j} \mathbb{E}[X_i X_j] - \mathbb{E}[X]^2. \\ =&\binom{d^{(s)}}{2} \frac{1 + (1 - 4/d^{(s)})^{\psi} - 2(1 - 2/d^{(s)})^{\psi}}{2} \\ &+ \frac{d^{(s)}(1 - (1 - 2/d^{(s)})^{\psi})}{2} - \left(\frac{d^{(s)}(1 - (1 - 2/d^{(s)})^{\psi})}{2}\right)^2.\end{aligned}$$

After simplifying the above expression we have

$$d^{(s)^2} \frac{(1 - 4/d^{(s)})^{\psi} - (1 - 2/d^{(s)})^{2\psi}}{4} + d^{(s)} \frac{1 - (1 - 4/d^{(s)})^{\psi}}{4}.$$

There are two terms in the above expression. In the first term, the numerator (both terms) of the coefficient of ${d^{(s)}}^2$ asymptotically converges to $e^{-\frac{4\psi}{d^{(s)}}}$. A careful analysis can show that when $\psi = d^{(s)}\frac{1-(1-2/d^{(s)})^{\psi}}{2}$, the term $e^{-\frac{4\psi}{d^{(s)}}}$ asymptotically converges to $O(1/{d^{(s)}}^2)$. Thus, the first term asymptotically converges to $O(1)$. In the second term, the numerator term of the coefficient of $d^{(s)}$ asymptotically converges to $1-e^{-\frac{4\psi}{d^{(s)}}}$. Thus, the second term asymptotically converges to $O(d^{(s)})$. Therefore, the variance of our estimate is $O(d^{(s)})$.

Accuracy of the estimate. Let us recall Equation 4.

$$\widetilde{\text{Ham}}(\mathbf{u}_s, \mathbf{v}_s) = -\frac{d^{(s)}}{2}\ln\left(1 - \frac{2\text{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})}{d^{(s)}}\right).$$

The value of $\widetilde{\text{Ham}}(\mathbf{u}_s, \mathbf{v}_s)$ determines the estimate of Cosine similarity between original vectors using Equation 5. The accuracy of this term depends on the correct estimate of the term $\text{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})/d^{(s)}$. There are two possibilities of inaccuracies in the expression mentioned above. The first one is due to the approximation of the bin-ball analysis with Poisson distribution in the above concentration bounds. However, we argue that this is not significant since the expected value of the estimate $\frac{\text{Ham}(\mathbf{u^{(s)}},\mathbf{v^{(s)}})}{d^{(s)}}$ as per bin-ball analysis is $\frac{1-(1-2/d^{(s)})^{\text{Ham}(\mathbf{u}_s,\mathbf{v}_s)}}{2}$, and as per Poisson distribution it is $1-e^{-\frac{2\text{Ham}(\mathbf{u}_s,\mathbf{v}_s)}{d^{(s)}}}$. It is clear that the difference between these two estimates is $O(1)$ and is not significant. The second source of inaccuracy is when the actual value of the estimate deviates from its expected value. Especially when the value of the estimate gets closer to $1/2$, the value of $\widetilde{\text{Ham}}(\mathbf{u}_s, \mathbf{v}_s)$ tends to infinity, which gives unfavorable results. Thus, we would like to have the expected value of the estimate (due to Poisson approximation) less than $1/2$. For lower Cosine similarity thresholds, the value of the estimate becomes closer to $1/2$ as the value of $\text{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})$ becomes high. As a consequence, a higher value of $d^{(s)}$ is required to obtain a desired performance. For higher Cosine similarity thresholds, the value $\text{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})$ is relatively smaller which leads to the value of the estimate being less than $1/2$ and producing encouraging results. This behavior has also been reflected in our experimental study.

Please note that our proposed algorithm Simsketch can also be used in any other application which requires compressing binary data while maintaining Hamming distance between data points. Furthermore, similar to Simhash, it can also be used in scaling up the performance of another popular hashing algorithm – WTA (winner take all) (Yagnik et al., 2011) – a sparse embedding method that transforms the input feature space into binary vectors such that Hamming distance in the resulting space closely correlates with *rank similarity measures*.

4. Empirical evaluation

Hardware description. Amazon Machine Image (AMI) : Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type Instance Type:- m4.2xlarge (26 ECUs, 8 vCPUs, 2.4 GHz, Intel Xeon E5-2676v3, 32 GiB memory, EBS only). In order to reduce the effect of randomness, we repeated each experiment several times and took the mean. No special optimisation techniques were adopted in our implementations.

Datasets. The experiments were performed on publicly available datasets - namely, BBC News Datasets (number of points = 2225, dimension = 9635) (Greene and Cunningham, 2006), NYTimes news articles (number of points = 300000, dimension = 102660), Enron Emails (number of points = 39861, dimension = 28102), and KOS blog entries (number of points = 3430, dimension = 6906) from the UCI machine learning repository (Lichman, 2013). We considered the entire corpus of KOS and BBC News, while for NYTimes and ENRON we took a sample of size 5000.

4.1. Experiment 1: Accuracy of Estimation

In this set of experiment, we evaluated the fidelity of the estimate of our proposed sketching algorithm, in the case when data points were highly similar. We discuss it as follows.

Evaluation Metric. In order to understand the behavior of Simsketch when the data points are highly similar, we need to extract samples of highly similar pairs of points from the datasets. To this purpose, we iterated over all the pairs, and extracted those whose Cosine similarities were higher than the given thresholds $\in \{0.95, 0.9, 0.85, 0.8\}$. For example: for the threshold value 0.95, we considered only those pairs whose Cosine similarities are higher than 0.95. Thus, we generated datasets in which most of the points were highly similar. We used mean square error (MSE) as our evaluation criteria. We first compressed the pruned datasets using Simhash at $d = 10$K. We emphasize that sketches of size $d = 10$K are space-efficient representations of the given datasets because originally they are represented in real-valued vectors while after applying Simhash they get mapped into binary vectors. Then using Simsketch we further compressed the sketch obtained by Simhash to various values of $d^{(s)}$. In order to compare the performance of Simsketch with Simhash, we compressed the pruned datasets solely using Simhash to the same dimension of $d^{(s)}$. For every pair of data points, we calculated the square of the difference between their estimated Cosine similarity after the result of Simsketch, and the corresponding ground truth Cosine similarity. We added these values for all such pairs and calculated its average. The value of this quantity is at most 1, so we computed the negative logarithm base e of this quantity. A smaller MSE corresponds to a larger $-\log(\text{MSE})$, therefore, a higher value of $-\log(\text{MSE})$ is an indication of better performance. We computed these $-\log(\text{MSE})$ values on various values of $d^{(s)}$, and compared them with the corresponding values obtained via Simhash.

Insights. We summarize our results in Figures 2,3,4. Our experiment can be thought of as a two-step process. First, the application of Simhash on real-valued vectors and second, application of Simsketch on the binary vectors obtained post Simhash. We quantify the error associated with Simsketch $w.r.t$ that obtained via Simhash at $d = 10$K. In our experiments, on higher and intermediate thresholds values, the error (MSE) associated is significantly less, that is the $-\log(\text{MSE})$ is more when contrasted to Simhash at $d^{(s)}$. High threshold values such as $\{0.95 \ldots 0.75\}$ indicate that data points are highly similar, which further imply that the Hamming distance between the binary vectors obtained after Simhash are small. Therefore, due to Equation 4, higher values of $-\log(\text{MSE})$ are achieved even on small values of $d^{(s)}$. Furthermore, on intermediate threshold values, Hamming distance between the binary vectors obtained after Simhash tends to become high, therefore a higher value of $d^{(s)}$ is required to achieve a good performance. This proves that Simsketch offers a more robust measure of the Cosine similarity.

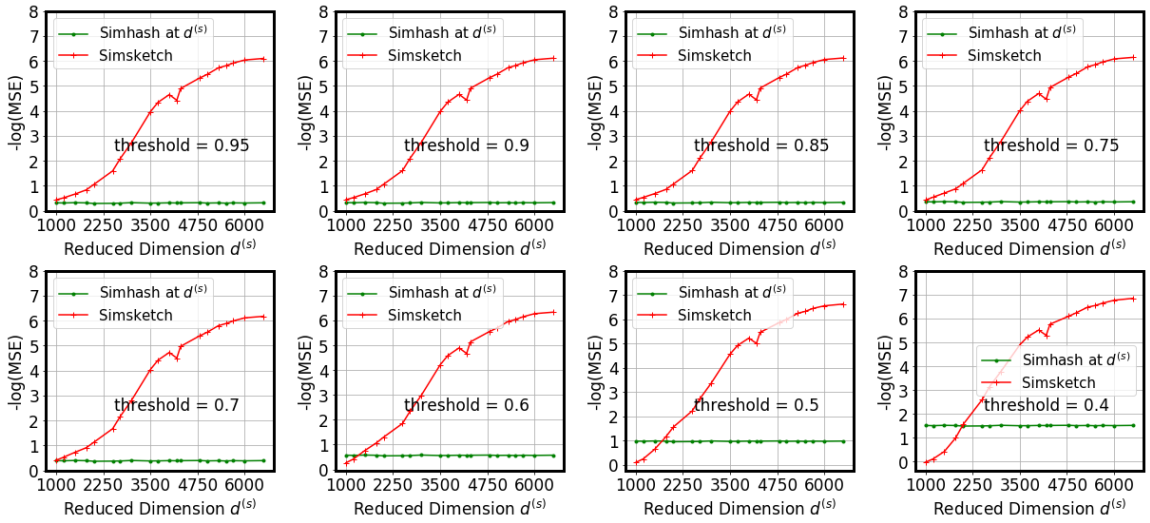

Figure 2: Comparison of the $-\log(\text{MSE})$ of Simsketch and Simhash at various values of $d^{(s)}$ on NYTimes. Simsketch is applied to the sketch obtained after applying Simhash on the original dataset with $d = 10\text{K}$. Recall that a higher $-\log(\text{MSE})$ corresponds to a lower MSE, which is an indication of an accurate compression.

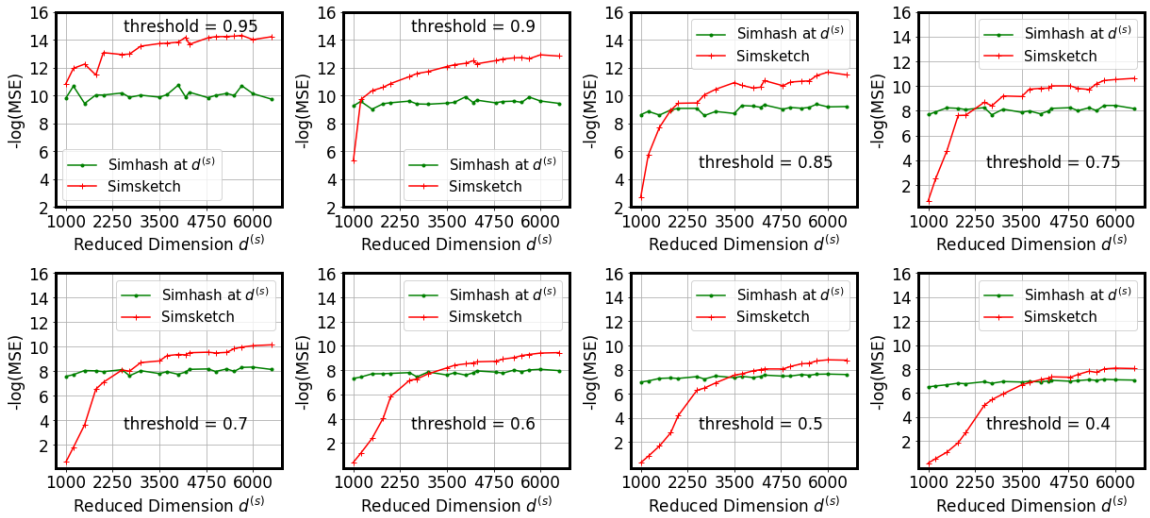

Figure 3: Comparison of the $-\log(\text{MSE})$ of Simsketch and Simhash at various values of $d^{(s)}$ on ENRON dataset. Simsketch is applied to the sketch obtained after applying Simhash on the original dataset with $d = 10\text{K}$.

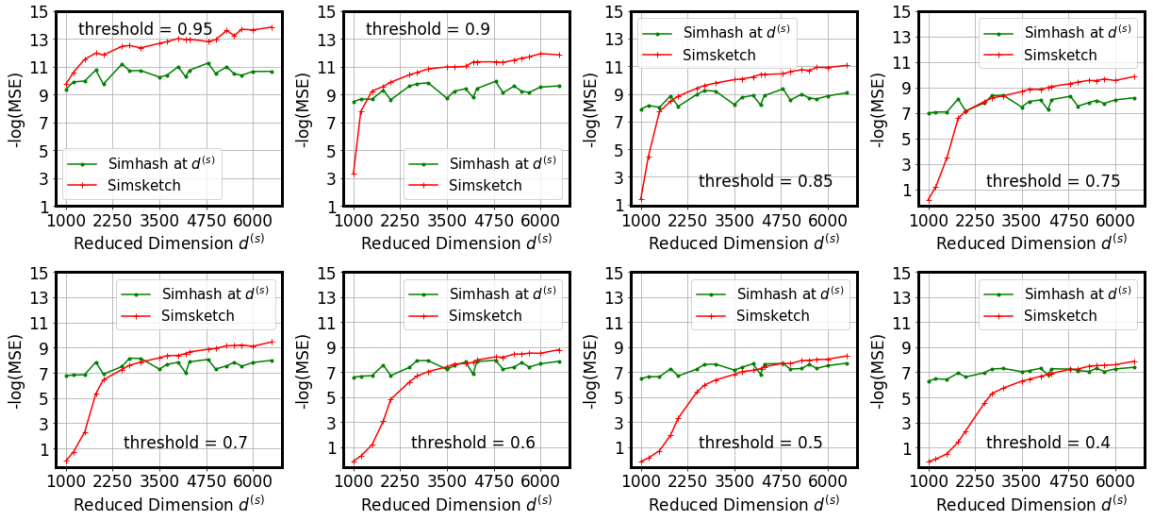

Figure 4: Comparison of the $-\log(\text{MSE})$ of Simsketch and Simhash at various values of $d^{(s)}$ on KOS dataset. Simsketch is applied to the sketch obtained after applying Simhash on the original dataset with $d = 10\text{K}$.

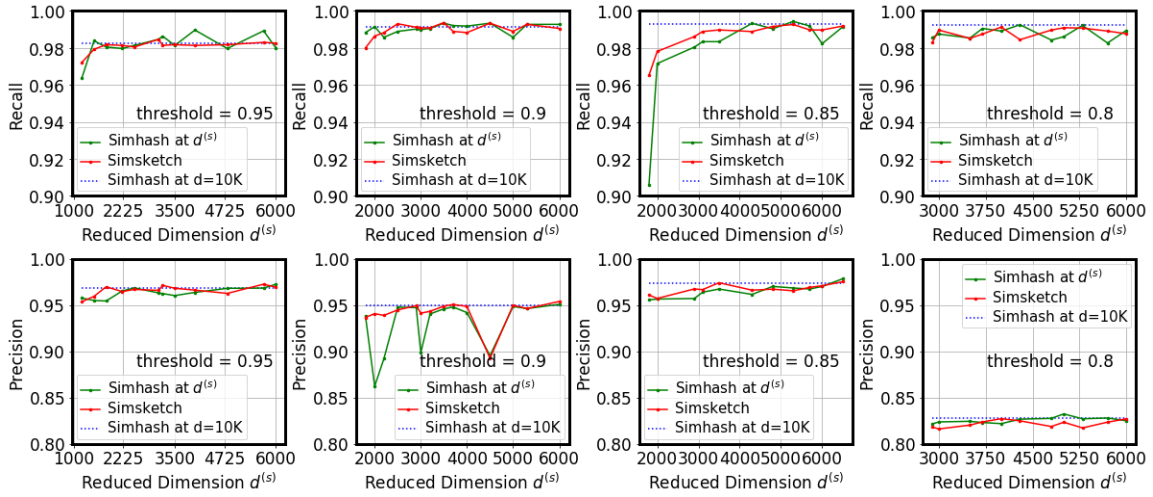

Figure 5: Comparison of the precision-recall between Simsketch and Simhash with thresholds= $\{0.95, \ldots, 0.8\}$ on ENRON.

4.2. Experiment 2: All-pair-similarity search

Evaluation Metric. Here, we used the *all-pair-similarity* search as our evaluation task, which is a fundamental subroutine in several fundamental data mining applications such as query refinement for search engines, semantic similarity in text (Reimers and Gurevych, 2019) collaborative filtering (Sarwar et al., 2001), near-duplicate detection (Bayardo et al.,

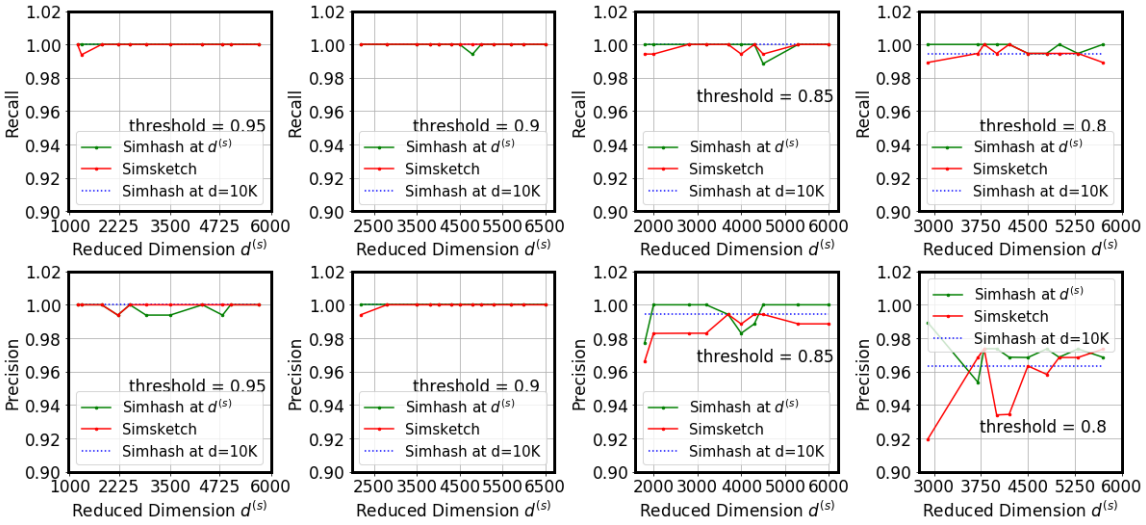

Figure 6: Comparison of the precision-recall between Simsketch and Simhash with thresholds $= \{0.95, 0.9, 0.85, 0.8\}$ on BBC.

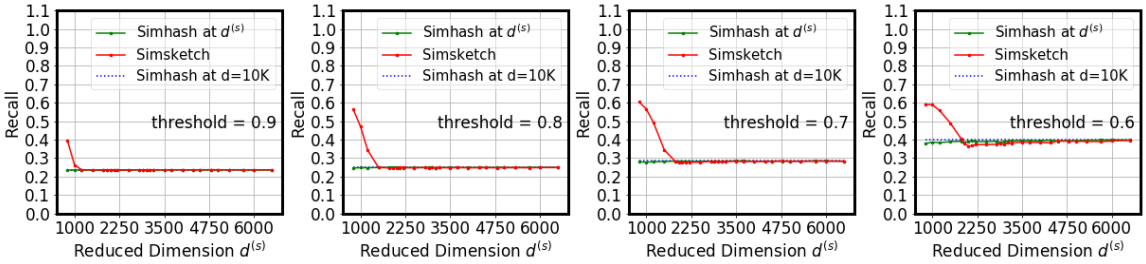

Figure 7: Comparison of the recall between Simsketch and Simhash with thresholds$=$ $\{0.9, \ldots, 0.6\}$ on NYTimes.

2007). As Simhash is a popular choice in such applications, Simsketch can further offer not only a more succinct sketch but also a scaled up performance.

For this set of experiment, we first calculated the ground truth result by retrieving every pair of data points whose Cosine similarity was higher than threshold values $\in \{0.95, \ldots, 0.3\}$. We compressed the dataset to the dimension $d = 10\text{K}$ using Simhash. We further compressed the sketch obtained after Simhash using Simsketch to various values of $d^{(s)}$. We compared the performance of Simsketch with Simhash algorithm. In order to do so, we compressed the original dataset using Simhash to the same values of $d^{(s)}$ and evaluated its performance. We used the *precision-recall* ratio as our standard measure. We define it as follows. If the set $\mathcal{O}$ denotes the ground truth result (result on the uncompressed dataset), and the set $\mathcal{O}'$ denotes Simhash/Simsketch result, then precision $:= |\mathcal{O} \cap \mathcal{O}'|/|\mathcal{O}'|$ and recall $:= |\mathcal{O} \cap \mathcal{O}'|/|\mathcal{O}|$.

Insights. We summarize our results in Figures 7,5,6 for the higher values of Cosine similarity threshold. It is easy to verify that even on smaller values of $d^{(s)}$, the precision-recall performance of Simsketch is almost equivalent to that of Simhash at $d = 10$K. For example: for 0.95 threshold even for $d^{(s)} = 1000$, we obtained almost the same *precision-recall* as of Simhash at $d = 10$K. Thus, we were able to reduce the sketch size $10\times$ while offering almost the same *precision-recall* values. We further compared and contrasted the performance of our proposed algorithm Simsketch with Simhash. It is easy to verify that Simsketch significantly outperforms Simhash for higher threshold values on the *recall* measure, especially for high dimensional dataset such as NYTimes (see Figure 7). Further, on the *precision* measure, for higher threshold values such as 0.95 and 0.9 Simsketch significantly outperforms Simhash, whereas for not so high threshold values such as 0.85 and 0.8, the performance of Simsketch is somewhat comparable *w.r.t.* Simhash. Simsketch gave better results for certain initial and intermediate values of $d^{(s)}$.

Performance of Simsketch **on low-threshold values.** We argue that our result holds true irrespective of similarity thresholds. However, on lower threshold values, a higher value of $d^{(s)}$ is required to achieve a desired performance. On lower threshold values, the value of term $\text{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})$ is high, thus, a large value of $d^{(s)}$ is required to bring down the value of the term $2\text{Ham}(\mathbf{u^{(s)}}, \mathbf{v^{(s)}})/d^{(s)}$ less than $1/2$ (Equation 4). This is necessary to find a correct estimate of the Cosine similarity (Equation 5). We performed experiments on low threshold values for $-\log(\text{MSE})$ and summarized our results in Figure 8.

We found that for low threshold values, the performance of Simsketch is somewhat moderate when compared to its performance on higher threshold values. This behavior is in coherence with the arguments mentioned above which suggest that for a lower threshold value, a higher value of $d^{(s)}$ is required in order to achieve a desired performance.

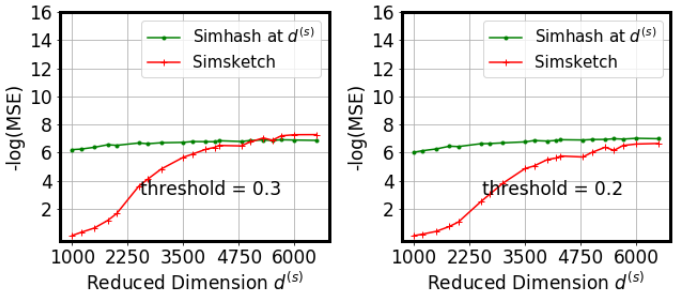

Figure 8: Comparison of $-\log(-\text{MSE})$ between Simsketch and Simhash on similarity threshold=$\{0.3, 0.2\}$ on ENRON.

Efficiency of Simsketch**.** We comment on the efficiency of Simsketch and summarize our results on ENRON and BBC News datasets in Figure 9. We noted the time required to compress the original dataset using Simhash at $d = 10$K, and that required by Simsketch to further compress this sketch to various values of $d^{(s)}$. We also noted the time required by Simhash to compress the original dataset on various values of $d^{(s)}$. We notice that the

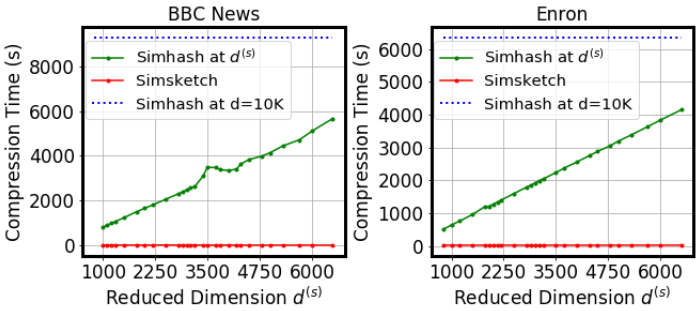

Figure 9: Comparison on the compression time between Simsketch and Simhash on ENRON and BBC News.

time required by Simsketch is negligible for all values of $d^{(s)}$ and on both the datasets, while compression time of Simhash grows linearly with $d^{(s)}$.

To summarize, Simsketch offers an efficient dimensionality reduction/sketching algorithm, which compresses a d-dimensional binary sketch obtained *via* Simhash to a $d^{(s)}$-dimensional binary sketch, where $d^{(s)} \ll d$. Simultaneously, on higher and intermediate threshold values, Simsketch preserves the desired performance of the d-dimensional sketch of Simhash on the MSE measure, and performs significantly better than the $d^{(s)}$-dimensional sketch of Simhash.

Recently, there are some results (Pratap et al., 2019, 2018c,b) which suggest compressing high-dimensional binary vectors to low-dimensional binary vectors such that the low-dimensional vectors closely estimate the Hamming distance of the full-dimensional data points. It will be interesting to see how these results compare with Simsketch.

5. Applications of Simsketch

As mentioned earlier, when the data dimension is very large, and data points have high Cosine similarity, Simhash requires a large dimensional sketch to accurately estimate the Cosine similarity between the data points. Such high dimensional sketches of Simhash could be impractical in several applications. In such scenarios Simsketch can be a viable substitute of Simhash by offering succinct and accurate sketches of the high dimensional data points. In what follows we mention several fundamental applications where Simhash is currently in use. We can potentially use Simsketch when data points are of high dimensional and share high similarity between one another. We defer a detailed comparison on the advantage of Simsketch on such applications to the full version of the paper.

Faster and scalable ranking of documents. Given a corpus of documents and a set of query documents, the task is to find all documents in the corpus that are similar to the query documents. This problem is a fundamental sub-routine in many applications like near-duplicate data detection (Manku et al., 2007; Sood and Loguinov, 2011), efficient document similarity search (Jiang and Sun, 2011; Nogueira et al., 2019) plagiarism detection (Buyrukbilen and Bakiras, 2013). Simhash is a popular choice of algorithm for such

problems. However, when documents are very similar, Simsketch offers a more succinct sketch and helps in scaling up the performing of these algorithms.

Scalable Clustering of documents. Spherical k-means (Dhillon and Modha, 2001) is a popular choice of clustering text documents (Endo and Miyamoto, 2015; Pratap et al., 2018a). Typically these documents are represented as high dimensional and sparse vectors. In the case of high document similarity, Simsketch can be used to compress the documents into binary vectors, and on this compressed representation, clustering can be performed leading to efficient, scalable, and accurate clustering performance when compared to the corresponding clustering results on the original dataset.

Other Applications. Apart from the above applications, Simhash compression has been widely used in collaborative filtering (Bachrach et al., 2009; Sarwar et al., 2001), approximate nearest neighbor search (Charikar, 2002), compressing social networks (Chierichetti et al., 2009), all pair similarity (Bayardo et al., 2007). As the data objects are highly similar in most of these cases, Simsketch can help in scaling up the performance of these algorithms by offering a more compact binary sketch.

References

Alekh Agarwal, Oliveier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15:1111–1133, 2014. URL http://jmlr.org/papers/v15/agarwal14a.html.

Yoram Bachrach, Ely Porat, and Jeffrey S. Rosenschein. Sketching techniques for collaborative filtering. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2016–2021, 2009. URL http://ijcai.org/Proceedings/09/Papers/332.pdf.

Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 131–140, 2007. doi: 10.1145/1242572.1242591. URL http://doi.acm.org/10.1145/1242572.1242591.

Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 327–336, 1998. doi: 10.1145/276698.276781. URL http://doi.acm.org/10.1145/276698.276781.

Sahin Buyrukbilen and Spiridon Bakiras. Secure similar document detection with simhash. In *Secure Data Management - 10th VLDB Workshop, SDM 2013, Trento, Italy, August 30, 2013, Proceedings*, pages 61–75, 2013. doi: 10.1007/978-3-319-06811-4_12. URL https://doi.org/10.1007/978-3-319-06811-4_12.

Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal,*

Québec, Canada, pages 380–388, 2002. doi: 10.1145/509907.509965. URL http://doi.acm.org/10.1145/509907.509965.

Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 219–228, 2009. doi: 10.1145/1557019.1557049. URL http://doi.acm.org/10.1145/1557019.1557049.

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001. doi: 10.1023/A:1007612920971. URL https://doi.org/10.1023/A:1007612920971.

Yasunori Endo and Sadaaki Miyamoto. Spherical k-means++ clustering. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 103–114. Springer, 2015.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 518–529, 1999. URL http://www.vldb.org/conf/1999/P49.pdf.

Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42 (6):1115–1145, 1995. doi: 10.1145/227683.227684. URL http://doi.acm.org/10.1145/227683.227684.

Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press, 2006.

Qixia Jiang and Maosong Sun. Semi-supervised simhash for efficient document similarity search. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 93–101, 2011. URL http://www.aclweb.org/anthology/P11-1010.

Norman L. Johnson and Samuel Kotz. *Urn models and their application: An approach to modern discrete probability theory*. John Wiley and Sons, New York; Chichester, 1977.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 141–150, 2007. doi: 10.1145/1242572.1242592. URL http://doi.acm.org/10.1145/1242572.1242592.

Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005. ISBN 978-0-521-83540-4.

Michael Mitzenmacher, Rasmus Pagh, and Ninh Pham. Efficient estimation for high similarities using odd sketches. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 109–118, 2014. doi: 10.1145/2566486.2568017. URL http://doi.acm.org/10.1145/2566486.2568017.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

Rameshwar Pratap, Anup Anand Deshmukh, Pratheeksha Nair, and Tarun Dutt. A faster sampling algorithm for spherical k-means. In *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14-16, 2018*, pages 343–358, 2018a. URL http://proceedings.mlr.press/v95/pratap18a.html.

Rameshwar Pratap, Raghav Kulkarni, and Ishan Sohony. Efficient dimensionality reduction for sparse binary data. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 152–157, 2018b. doi: 10.1109/BigData.2018.8622338. URL https://doi.org/10.1109/BigData.2018.8622338.

Rameshwar Pratap, Ishan Sohony, and Raghav Kulkarni. Efficient compression technique for sparse sets. In *Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III*, pages 164–176, 2018c. doi: 10.1007/978-3-319-93040-4_14. URL https://doi.org/10.1007/978-3-319-93040-4_14.

Rameshwar Pratap, Debajyoti Bera, and Karthik Revanuru. Efficient sketching algorithm for sparse binary data. In *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pages 508–517, 2019. doi: 10.1109/ICDM.2019.00061. URL https://doi.org/10.1109/ICDM.2019.00061.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

Sadhan Sood and Dmitri Loguinov. Probabilistic near-duplicate detection using simhash. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1117–1126, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063737. URL http://doi.acm.org/10.1145/2063576.2063737.

Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-Sung Lin. The power of comparative reasoning. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2431–2438, 2011. doi: 10.1109/ICCV.2011.6126527. URL https://doi.org/10.1109/ICCV.2011.6126527.