Efficient Sketching Algorithm for Sparse Binary Data

Rameshwar Pratap   Debajyoti Bera   Karthik Revanuru

***Abstract*—Recent advancement of the WWW, IOT, social network, e-commerce, etc. have generated a large volume of data. These datasets are mostly represented by high dimensional and sparse datasets. Many fundamental subroutines of common data analytic tasks such as clustering, classification, ranking, nearest neighbor search, etc. scale poorly with the dimension of the dataset. In this work, we address this problem and propose a sketching (alternatively, dimensionality reduction) algorithm – $\mathrm{BinSketch}$ (Binary Data Sketch) – for sparse binary datasets. $\mathrm{BinSketch}$ preserves the binary version of the dataset after sketching and maintains estimates for multiple similarity measures such as Jaccard, Similarity, Inner-Product similarities, and Hamming Distance, on the same sketch. We present a theoretical analysis of our algorithm and complement it with extensive experimentation on several real-world datasets. We compare the performance of our algorithm with the state-of-the-art algorithms on the task of mean-square-error and ranking. Our proposed algorithm offers a comparable accuracy while suggesting a significant speedup in the dimensionality reduction time, with respect to the other candidate algorithms. Our proposal is simple, easy to implement, and therefore can be adopted in practice.**

I. INTRODUCTION

Due to technological advancement, recent year have witnessed a dramatic increase in our ability to collect data from various sources like WWW, IOT, social media platforms, mobile applications, finance, and biology. For example, in many web applications, the volume of datasets are of the terascale order, with trillions of features [1]. The high dimensionality incurs high memory requirements and computational cost during the training. Further, most of such high dimensional datasets are sparse, owing to a wide adaption of "Bag-of-words" (BoW) representations. For example: in the case of document representation, word frequency within a document follows power law – most of the words occur rarely in a document, and higher order shingles occur only once. We focus on the binary representation of the datasets which is quite common in several applications [9], [10], [16], [20].

Measuring similarity score of data points under various similarity measures is a fundamental subroutine in several applications such as clustering, classification, identifying nearest neighbors, ranking, and it play an important role in various data mining, machine learning, and information retrieval tasks. However, due to the "curse of dimensionality" a brute-force way of computing the similarity scores in the high dimensional dataset is in-feasible, and at times impossible. In this work, we address this question and propose an efficient dimensionality reduction algorithm for sparse binary datasets that generates a succinct sketch of the dataset while preserving estimates for computing the similarity score between data objects.

A. Our Contribution

We first informally describe our sketching algorithm.

BinSketch: (**Bin**ary Data **Sketch**ing) Given a d-dimensional binary vector $a \in \{0,1\}^d$, our algorithm reduces it to a N-dimensional binary vector $a_s \in \{0,1\}^{\mathrm{N}}$, where N is specified later. It randomly maps each bit position (say) $\{i\}_{i=1}^{d}$ to an integer $\{j\}_{j=1}^{\mathrm{N}}$. To compute the j-th bit of a_s, it checks which bit positions have been mapped to j, computes the $\mathtt{bitwise}-\mathtt{OR}$ of the bits located at those positions and assigns it to $a_s[j]$.

A simple and exact solution to the problem is to represent each binary vector by a (sorted) list (or vector) of the indices with value one. In this representation, the space required in storing a vector is $O(\psi \log d)$ bits – as we need $O(\log d)$ bits for storing each index, and there are at most ψ indices with non-zero value. Further, the time complexity of computing the (say) inner product of two originally ψ-sparse binary vectors is $O(\psi \log d)$. Therefore, both the storage as well as the time complexity of calculating similarity depend on the original dimension d and does not scale for large values of d. For high dimensional sparse binary data, we show how to construct highly compressed binary sketches whose length depends only on the data sparsity. Furthermore, we present techniques to compute similarity between vectors from their sketches alone. Our main technique is presented in Algorithm 1 for inner product similarity and the following theorem summarizes it.

Theorem 1 (Estimation of inner product)**.** *Suppose we want to estimate the* Inner Product *of d-dimensional binary vectors, whose sparsity is at most ψ, with probability at least $1-\rho$. We can use* $\mathrm{BinSketch}$ *to construct* N*-dimensional binary sketches where $\mathrm{N} = \psi\sqrt{\frac{\psi}{2}\ln\frac{2}{\rho}}$. If a_s and b_s denote the sketches of vectors a and b, respectively, then* $\mathrm{IP}(a,b)$ *can be estimated with accuracy $O(\sqrt{\psi \ln \frac{6}{\rho}})$ using Algorithm 1.*

We also present Algorithm 2 for estimating Hamming Distance, Algorithm 3 for estimating Jaccard Similarity and Algorithm 4 for estimating Cosine Similarity; all these algorithms are designed based on Algorithm 1 and so follow similar accuracy guarantees.

Extension for categorical data compression. Our result can be easily extended for compressing Categorical datasets. The categorical dataset consists of several categorical features.

Examples of categorical features are sex, weather, days in a week, age group, educational level, etc. We consider a type of Hamming distance for defining the distance between two categorical data points. For two d dimensional categorical data points u and v, the distance between them is defined as follows: $\mathrm{D}(u,v) = \Sigma_{i=1}^{d} \mathrm{dist}(u[i], v[i])$, where

$$\mathrm{dist}(u[i], v[i]) = \begin{cases} 1, & \text{if } u[i] \neq v[i], \\ 0, & \text{otherwise.} \end{cases}$$

In order to use $\mathrm{BinSketch}$, we need to preprocess the datasets. We first encode categorical feature via *label-encoding* followed by *one-hot-encoding*. In the label encoding step, features are encoded as integers. For a given feature, if it has m possible values, we encode them with integers between 0 and $m-1$. In one-hot-encoding step, we convert the feature value into a m length binary string, where 1 is located at the position corresponding to the result of the label-encoding step. [1] This preprocessing convert categorical dataset to a binary dataset. Please note that after preprocessing Hamming distance between the binary version of the data points is equal to the corresponding categorical distance $\mathrm{D}(,)$, stated above. We can now compress the binary version of the dataset using $\mathrm{BinSketch}$ and due to Algorithm 2, the compressed representation maintains the Hamming distance.

In Section III we present the proof of Theorem 1 where we explain the theoretical reasons behind the effectiveness of $\mathrm{BinSketch}$. As is usually the case for hash functions, practical performance often outshines theoretical bounds; so we conduct numerous experiments on public datasets. Based on our experiment results reported in Section IV we make the claim that $\mathrm{BinSketch}$ is the best option for compressing sparse binary vectors while retaining similarity for many of the commonly used measures. The accuracy obtained is comparable with the state-of-the-art sketching algorithms, especially at high similarity regions, while taking almost negligible time compared to similar sketching algorithms proposed so far.

B. Related work

Our proposed algorithm is very similar in nature to the BCS algorithm [25], [26], which suggests a randomized bucketing algorithm where each index of the input is randomly assigned to one of the $O(\psi^2)$ buckets; ψ denotes the sparsity of the dataset. The sketch of an input vector is obtained by computing the parity of the bits fallen in each bucket. We offer a better compression bound than theirs. For a pair of vectors, their compression bounds are $O(\psi^2)$, while ours is $O(\psi\sqrt{\psi})$. This is also reflected in our empirical evaluations, on small values of compression length, we outperform as compare to their algorithms. However, the compression times (or dimensionality reduction time) of both the algorithms are somewhat comparable.

For Jaccard Similarity, we compare the performance of our algorithms with $\mathrm{MinHash}$ [5], DOPH [27] – a faster variant of $\mathrm{MinHash}$, and $\mathrm{OddSketch}$ [23]. We would like to point out some key differences between $\mathrm{OddSketch}$ and $\mathrm{BinSketch}$. $\mathrm{OddSketch}$ is two-step in nature that takes the sketch obtained by running $\mathrm{MinHash}$ on the original data as input, and outputs binary sketch which maintains an estimate of the original Jaccard similarity. Due to this two-step nature, its compression time is higher (see Table I and Figure 3). The number of $\mathrm{MinHash}$ functions used in $\mathrm{OddSketch}$ (denoted by k) is a crucial parameter and the authors suggested using k such that the pairwise symmetric difference is approximately $\mathrm{N}/2$. Empirically they suggest using $k = \mathrm{N}/(4(1-J))$, where J is the similarity threshold. We argue that not only tuning k is an important step but it is unclear how this condition will be satisfied for a diverse dataset, on the contrary, $\mathrm{BinSketch}$ requires no such parameter. Furthermore, $\mathrm{OddSketch}$ doesn't provide any closed form expression to estimate accuracy and confidence. However, the variance of the critical term of their estimator is linear in the size of the sketch, *i.e.* N. Whereas our confidence interval is of the order of $\sqrt{\psi}$ which could be far smaller compared to N, even for non-sparse data. Finally, compared to the Poisson approximation based analysis used in $\mathrm{OddSketch}$, we employed a tighter martingale-based analysis leading to (slightly) better concentration bounds (compare, e.g., the concentration bounds for estimating the size of a set from its sketch).

For Cosine Similarity, we compare $\mathrm{BinSketch}$ with $\mathrm{SimHash}$ [11], CBE [31] – a faster variant of $\mathrm{SimHash}$, $\mathrm{MinHash}$ [29], DOPH on the sketch obtained by $\mathrm{MinHash}$ [29]. For the Inner Product, BCS [26], Asymmetric MinHash [29], and Asymmetric DOPH – DOPH [27] on the sketch obtained by [29], were the competing algorithms. In all these similarity measures, for sparse binary datasets, our proposed algorithm is faster, while simultaneously offering almost a similar performance as compared to the baselines. We experimentally compare the performance on several real-world datasets and observed the results that are in line with these observations. Further, in order to get a sketch of size N, our algorithm requires a lesser number of random bits, and require only one pass to the datasets. These are the major reasons due to which we obtained good speedup in compression time. We summarize this comparison in Table I. Finally, a major advantage of our algorithm, similar to [25], [26], is that it gives one-shot sketching by maintaining estimates of multiple similarity measures in the same sketch; this is is contrast to usual sketches that are customized for a specific similarity.

a) Connection with Bloom Filter.: $\mathrm{BinSketch}$ appears structurally similar to a Bloom filter with one hash function. The standard Bloom filter is a space-efficient data-structure for *set-membership* queries; however, there is an alternative approach that can be used to estimate the intersection between two sets [6]. However, it is unclear how estimates for other similarity measures can be obtained. We answer this question positively and suggests estimates for all the four similarity measures in the same sketch. We also show that our estimates are strongly concentrated around their expected values.

[1]Both label-encoder and one-hot-encoder are available in `sklearn` as `labelEncoder` and `OneHotEncoder` packages.

TABLE I

A COMPARISON AMONG THE CANDIDATE ALGORITHMS, ON THE NUMBER OF RANDOM BITS AND THE COMPRESSION TIME, TO GET A SKETCH OF LENGTH N OF ONE DATA OBJECT. COMPRESSION TIME INCLUDES BOTH (I) TIME REQUIRED TO GENERATE HASH FUNCTION, WHICH IS OF ORDER THE NUMBER OF RANDOM BITS, (II) TIME REQUIRED TO GENERATE THE SKETCH USING THE HASH FUNCTIONS. THE PARAMETER k FOR OddSketch DENOTES THE NUMBER OF PERMUTATIONS REQUIRED BY AN INTERMEDIATE MinHash STEP.

Algorithm	No of random bits	Compression time
BinSketch	$O(d \log \mathrm{N})$	$O(d \log \mathrm{N} + \psi)$
BCS [25], [26]	$O(d \log \mathrm{N})$	$O(d \log \mathrm{N} + \psi)$
DOPH [27]	$O(d \log d)$	$O(d \log d + \psi + \mathrm{N})$
CBE [31]	$O(d)$	$O(d \log d)$
OddSketch [23]	$O(k(d \log d + \mathrm{N}))$	$O(k(d \log d + \mathrm{N} + \psi))$
SimHash [11]	$O(d\mathrm{N})$	$O((d + \psi)\mathrm{N})$
MinHash [5]	$O((d \log d)\mathrm{N})$	$O((d \log d + \psi)\mathrm{N})$

C. *Applicability of our results*

For high dimensional sparse binary datasets, BinSketch due to its simplicity, efficiency, and performance, can be used in numerous applications that require a sketch preserving Jaccard, cosine, Hamming distance or inner product similarity.

Scalable Ranking and deduplication of documents.: Given a corpus of documents and a set of query documents, a goal is to find all documents in the corpus that are "similar" to query documents under a given similarity measure (e.g., Jaccard, cosine, inner product). This problem is a fundamental sub-routine in many applications like near-duplicate data detection [4], [17], [22], [30], efficient document similarity search [19], [29], plagiarism detection [4], [7], etc. and dimensionality reduction is one way to address this problem. In Subsection IV-B we provide empirical validation that BinSketch offers significant speed-up in dimensionality reduction while offering a comparable accuracy.

Scalable Clustering of documents.: BinSketch can be used in scaling up the performance of several clustering algorithms, in the case of high-dimensional and sparse datasets. For instance, in the case of Spherical k-means clustering, which is the problem of clustering data points using Cosine Similarity, one can use [13], [24]; and for k-mode clustering, which is clustering using Hamming Distance, one can use k-mode [18], on the sketch obtained by BinSketch.

Other Applications.: Beyond the above-noted applications, sketching techniques have been used widely in application such as Spam detection [3], compressing social networks [12] all pair similarity [2], Frequent Itemset Mining [8]. As BinSketch offers significant speed-up in dimensionality reduction time and simultaneously provides a succinct and accurate sketch, it helps in scaling up the performance of the respective algorithms.

II. BACKGROUND

Notations	
N	dimension of the compressed data.
ψ	sparsity bound.
$u[i]$	i-th bit position of binary vector u.
$\|u\|$	number of 1's in the binary vector u.
$\mathrm{Cos}(u,v)$	Cosine similarity between u and v.
$\mathrm{JS}(u,v)$	Jaccard similarity between u and v.
$\mathrm{Ham}(u,v)$	Hamming distance between u and v.
$\mathrm{IP}(u,v)$	Inner product between u and v.

a) SimHash for cosine similarity [11], [14].: The cosine similarity between a pair of vectors $u, v \in \mathbb{R}^d$ is defined as $\langle u, v\rangle / \|u\|_2 \cdot \|v\|_2$. To compute a sketch of a vector u, SimHash [11] generates a random vector $r \in \{-1, +1\}^d$, with each component chosen uniformly at random from $\{-1, +1\}$ and a 1-bit sketch is computed as

$$\mathrm{SimHash}^{(r)}(u) = \begin{cases} 1, & \text{if } \langle u, r\rangle \geq 0. \\ 0, & \text{otherwise.} \end{cases}$$

SimHash was shown to preserve inner product in the following manner [14]. Let θ be an angle such that $\cos\theta = \langle u, v\rangle / \|u\| \cdot \|v\|$. Then,

$$\Pr[\mathrm{SimHash}^{(r)}(u) = \mathrm{SimHash}^{(r)}(v)] = 1 - \frac{\theta}{\pi},$$

b) MinHash for Jaccard and cosine similarity.: The Jaccard similarity between a pair of set $u, v \subseteq \{1, 2, \ldots d\}$ is defined as $\mathrm{JS}(u,v) = \frac{|u \cap v|}{|u \cup v|}$. Broder et al. [5] suggested an algorithm – MinHash – to compress a collection of sets while preserving the Jaccard similarity between any pair of sets. Their technique includes taking a random permutation of $\{1, 2, \ldots, d\}$ and assigning a value to each set which maps to minimum under that permutation.

Definition 2 (Minhash [5])**.** *Let π be a random permutation over $\{1, \ldots, d\}$, then for a set $u \subseteq \{1, \ldots d\}$ $h_\pi(u) = \arg\min_i \pi(i)$ for $i \in u$.*

It was then shown by Broder et al. [4], [5] that

$$\Pr[h_\pi(u) = h_\pi(v)] = \frac{|u \cap v|}{|u \cup v|}.$$

Exploiting a similarity between Jaccard similarity of sets and cosine similarity of binary vectors, it was shown how to use MinHash for constructing sketches for cosine similarity in the case of sparse binary data [28].

c) BCS for sparse binary data [25], [26].: For sparse binary dataset, BCS offers a sketching algorithm that simultaneously preserves Jaccard similarity, Hamming distance and inner product.

Definition 3 (BCS)**.** *Let N be the number of buckets. Choose a random mapping b from $\{1 \ldots d\}$ to $\{1, \ldots N\}$. Then a vector $u \in \{0,1\}^d$ is compressed to a vector $u_s \in \{0,1\}^{\mathrm{N}}$ as follows:*

$$u_s[j] = \sum_{i : b(i) = j} u[i] \pmod 2.$$

III. Analysis of BinSketch

Let a and b denote two binary vectors in d-dimension, and $|a|$, $|b|$ denotes the number of 1 in a and b. Let $a_s, b_s \in \{0,1\}^{\mathrm{N}}$ denote the compressed representation of a and b, where N denotes the compression length (or reduced dimension). In this section we will explain our sketching method BinSketch and give theoretical bounds on its efficacy.

Definition 4 (BinSketch). *Let π be a random mapping from $\{1,\dots d\}$ to $\{1,\dots \mathrm{N}\}$. Then a vector $a \in \{0,1\}^d$ is compressed into a vector $a_s \in \{0,1\}^{\mathrm{N}}$ as*

$$a_s[j] = \bigvee_{i:\pi(i)=j} a[i]$$

Constructing a BinSketch for a dataset involves first, generating a random mapping π, and second, hashing each vector in the dataset using π. There could be N^d possible mappings, so choosing π requires $O(\log(\mathrm{N}^d)) = O(d \log N)$ time and that many random bits. Hashing a vector a involves only looking at the non-zero bits in a and that step takes time $O(\psi)$ since $|a| \le \psi$. Both these costs compete favourably with the existing algorithms as tabulated in Table I.

A. Inner-product similarity

The sketches, a_s's do not quite "preserve" inner-product by themselves, but are related to the latter in the following sense. We will use n to denote $1 - \frac{1}{\mathrm{N}} \in (0,1)$; it will be helpful to note that $n \to 1$ as N increases.

Lemma 5.

1. $\mathbb{E}(|a_s|/\mathrm{N}) = (1 - n^{|a|})$
2. $\mathbb{E}(\langle a_s, b_s\rangle/\mathrm{N}) =$

$$(1-n^{|a|})(1-n^{|b|}) + n^{|a|+|b|}\left[\left(\frac{1}{n}\right)^{\langle a,b\rangle} - 1\right] =$$
$$1 - n^{|a|} - n^{|b|} + n^{|a|+|b|+\langle a,b\rangle}$$

Proof. It will be easier to identify $a \in \{0,1\}^d$ as a subset of $\{1,\dots d\}$. The j-th bit of a_s can be set only by some element in a which can happen with probability $(1-(1-\frac{1}{\mathrm{N}})^{|a|})$. The j-th bit of both a_s and b_s is set if it is set by some element in $a \cap b$, or if it is set simultaneously by some element in $a \setminus (a \cap b) = a \setminus b$ and by another element in $b \setminus (a \cap b)$. This translates to the following probability that some particular bit is set in both a_s and b_s.

$$\left(1 - n^{|a\cap b|}\right) + n^{|a\cap b|}\left(1 - n^{|a\setminus b|}\right)\left(1 - n^{|b\setminus a|}\right)$$
$$= 1 - n^{|a|} - n^{|b|} + n^{|a|+|b|-|a\cap b|}$$
$$= (1-n^{|a|})(1-n^{|b|}) + n^{|a|+|b|}\left(\frac{1}{n^{|a\cap b|}} - 1\right)$$

The lemma follows from the above probabilities using the linearity of expectation. □

Note that the above lemma allows us to express $\langle a,b\rangle$ as

$$\langle a,b\rangle = |a| + |b| - \tfrac{1}{\ln n}\ln\left(n^{|a|} + n^{|b|} + \frac{\mathbb{E}(\langle a_s, b_s\rangle)}{\mathrm{N}} - 1\right)$$

Algorithm 1 now explains how to use this result to approximately calculate $\langle a,b\rangle$ using their sketches a_s and b_s.

Algorithm 1 BinSketch estimation of $\mathrm{IP}(a,b)$

Input: Sketches a_s of a and b_s of b

1: Estimate $\mathbb{E}[|a_s|]$ as $n_{a_s} = |a_s|$, $\mathbb{E}[|b_s|]$ as $n_{b_s} = |b_s|$
2: Estimate $\mathbb{E}[\langle a_s, b_s\rangle]$ as $n_{a_s,b_s} = \langle a_s, b_s\rangle$
3: Approximate $|a|$ as $n_a = \ln(1 - \frac{n_{a_s}}{\mathrm{N}})/\ln(n)$ and $|b|$ as $n_b = \ln(1 - \frac{n_{b_s}}{\mathrm{N}})/\ln(n)$
4: **return** approximation of $\langle a,b\rangle$ as

$$n_{a,b} = n_a + n_b - \tfrac{1}{\ln n}\ln\left(n^{n_a} + n^{n_b} + \frac{n_{a_s,b_s}}{\mathrm{N}} - 1\right)$$

We will prove that Algorithm 1 estimates $\langle a,b\rangle$ with high accuracy and confidence if we use $\mathrm{N} = \psi\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$; δ can be set to any desired probability of error and we assume that the sparsity ψ is not too small, say at least 20. Our first result proves that the n_{a_s} estimated above is a good approximation of $\mathbb{E}[|a_s|]$; exactly identical result holds for b_s and n_{b_s} too.

Lemma 6. *With probability at least $1-\delta$, it holds that*

$$\left|n_{a_s} - \mathbb{E}[|a_s|]\right| < \sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$$

Proof. The proof of this lemma is a simple adaptation of the computation of the expected number of non-empty bins in a balls-and-bins experiment that is found in textbooks and done using Doob's martingale. Identify the random mapping $\pi(a)$, where the number of 1's in a is denoted by $|a|$, as throwing $|a|$ black balls (and $d - |a|$ "no"-balls), one-by-one, into N bins chosen uniformly at random. Supposing we only consider the black balls in the bins, then $a_s[j]$ is an indicator variable for the event that the j-th bin is non-empty and the number of non-empty bins can be shown to be concentrated around their expectation [2]. Since the number of non-empty bins correspond to $|a_s|$, this concentration bound can be directly applied for proving the lemma.

Let $\mathcal{E}$ denote the event in the statement of the lemma. Then,

$$\Pr[\bar{\mathcal{E}}] \le \Pr\left[\left||a_s| - \mathbb{E}\big[|a_s|\big]\right| \ge \sqrt{\frac{|a|}{2}\ln\frac{2}{\delta}}\right] \le \delta$$

where $|a| \le \psi$ is used for the first inequality and the stated bound, with $m = |a|$, is used for the second inequality. □

Similar, but more involved, approach can be used to prove that $n_{a_s,b_s} = \langle a_s, b_s\rangle$ is a good estimation of $\mathbb{E}[\langle a_s, b_s\rangle]$.

Lemma 7. *With probability at least $1-\delta$, it holds that*

$$\left|n_{a_s,b_s} - \mathbb{E}[\langle a_s, b_s\rangle]\right| < \sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$$

[2] Using F to denote the number of non-empty bins and m the number of balls, Azuma-Hoeffding inequality states that $\Pr\left[|F - \mathbb{E}[F]| \ge \lambda\right] \le 2\exp(-2\lambda^2/m)$ (see Probability and Computing, Mitzenmacher and Upfal, Cambridge Univ. Press).

Proof. For a given $a, b \in \{0,1\}^d$, lets partition $\{1, \dots d\}$ into parts C (consisting of positions at which both a and b are 1), D (positions at which a is 1 and b is 0), E (positions at which a is 0 and b is 1) and F (the rest). Any random mapping π can treated as throwing $|C|$ grey balls, $|D|$ white balls, $|E|$ black balls, and $d-|C|-|D|-|E|$ "no"-balls randomly into N bins. Suppose we say that a bin is "greyish" if it either contains some grey ball or both a white and a black ball. The number of common 1-bits in a_s and b_s (that is $n_{a_s,b_s} = \langle a_s, b_s\rangle$) is now equal to the number of greyish bins. Observe that when any ball lands in some bin, say j, the number of greyish bins either remains same or increases by 1; therefore, we can say that the count of the greyish bins satisfies Lipschitz condition. This allows us to apply Azuma-Hoeffding inequality as above and prove the lemma; we will also need the fact that the number of greyish bins is at most ψ. □

The next lemma allows us to claim that our estimation of $|a|$ is also within reasonable bounds. It should be noted that our sketches $|a_s|$ do not explicitly save the number of 1's in a, so it is necessary to compute this number from our sketches; furthermore, since this estimate is not used elsewhere, we do not mandate it to be an integer either.

Lemma 8. *With probability at least $1-\delta$, it holds that*

$$\Big||a| - n_a\Big| < \frac{4}{\psi \ln \frac{1}{n}} = 4\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$$

Proof. Based on Lemma 5 and Algorithm 1, $n^{|a|} - n^{n_a} = [n_{a_s} - \mathbb{E}(|a_s|)]/N$. For the proof we use the upper bound given in Lemma 6 that holds with probability at least $1-\delta$. We need a few results before proceeding that are based on the standard inequality $\ln(1-x) \le -x$ for $0 < x < 1$.

Observation 9. $\ln\frac{1}{n} \ge \frac{1}{N}$ *($\because \ln n = \ln(1-1/N) \le -\frac{1}{N}$)*

Observation 10. $n_a = \ln(1-\frac{n_{a_s}}{N})/\ln n \le \frac{n_{a_s}}{N}/\ln(\frac{1}{n})$. *Since $n_{a_s} \le N$, we get that $n_a \le N$.*

Observation 11. $n^{n_a} \ge \frac{1}{2}$ *(proved in Appendix A).*

We use these observations for proving two possible cases of the lemma. We will use the notation $\Delta = \Big|n_a - |a|\Big|$.

case (i) $|a| \le n_a$**:** In this case $\Delta = n_a - |a|$ and

$$n^{|a|} - n^{n_a} = [n_{a_s} - \mathbb{E}(|a_s|)]/N$$

For the R.H.S., $[n_{a_s} - \mathbb{E}(|a_s|)]/N \le 1/\psi$ by Lemma 6.
For the L.H.S., we can write $n^{|a|} - n^{n_a} = n^{|a|}(1-n^{n_a-|a|}) \ge n^{\psi}(1-n^{\Delta})$ as $|a| \le \psi$. Furthermore, $n^{\psi} = (1-\frac{1}{N})^{\psi} \ge 1 - \frac{\psi}{N} > \frac{1}{2}$ since $\frac{\psi}{N} = 1/\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}} < \frac{1}{2}$ for reasonable values of ψ and δ.
Combining the bounds above we get the inequality $\frac{1}{2}(1-n^{\Delta}) < 1/\psi$ that we will further process below.

case (ii) $n_a \le |a|$**:** In this case $\Delta = |a| - n_a$ and

$$n^{n_a} - n^{|a|} = [\mathbb{E}(|a_s|) - n_{a_s}]/N$$

As above, R.H.S. is at most $1/\psi$ using Lemma 6 and L.H.S. can be written as $n^{n_a}(1-n^{\Delta})$. Further using Observation 11 we get the inequality, $\frac{1}{2}(1-n^{\Delta}) \le 1/\psi$.

For both the above cases we obtained that $\frac{1}{2}(1-n^{\Delta}) \le 1/\psi$, i.e., $1 - n^{\Delta} \le 2/\psi$. This gives us that $\Delta \ln n \ge \ln(1-2/\psi) \ge \frac{-2/\psi}{1-2/\psi} = \frac{-2}{\psi-2}$ employing the known inequality $\ln(1+x) \ge \frac{x}{x+1}$ for any $x > -1$. Since $n \in (0,1)$, we get the desired upper bound $\Delta \le \frac{2}{\psi-2}\frac{1}{\ln\frac{1}{n}} \le \frac{4}{\psi \ln \frac{1}{n}}$ (since $\frac{\psi}{2} \le \psi - 2$ for $\psi \ge 4$) $\le 4\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$ (using Observation 11). □

Of course a similar result holds for $|b|$ and n_b as well. The next lemma similarly establishes the accuracy of our estimation of $\langle a, b\rangle$.

Lemma 12. *With probability at least $1-3\delta$, it holds that*

$$\Big|\langle a,b\rangle - n_{a,b}\Big| < 14\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$$

We get the following from Algorithm 1 and Lemma 5.

$$\langle a,b\rangle = |a| + |b| + \frac{1}{\ln\frac{1}{n}}\ln\left[n^{|a|} + n^{|b|} + \frac{\mathbb{E}[\langle a_s, b_s\rangle]}{\mathrm{N}} - 1\right]$$
$$n_{a,b} = n_a + n_b + \frac{1}{\ln\frac{1}{n}}\ln\left(n^{n_a} + n^{n_b} + \frac{n_{a_s,b_s}}{\mathrm{N}} - 1\right)$$

in which $|a| \approx n_a$ (Lemma 8), $|b| \approx n_b$ (similarly), and $\mathbb{E}[\langle a_s, b_s\rangle] \approx n_{a_s,b_s}$ (Lemma 7), each happening with probability at least $1-\delta$. The complete proof that $n_{a,b}$ is a good approximation of $\langle a,b\rangle$ is mostly algebraic analysis of the above facts and is included in Appendix B.

Theorem 1 is a direct consequence of Lemma 12 for reasonably large ψ (say, beyond 20) and small δ (say, less than 0.1).

B. Hamming distance

The Hamming distance and the inner product similarity of two binary vectors a and b are related as

$$\mathrm{Ham}(a,b) = |a| + |b| - \mathrm{IP}(a,b)$$

The technique used in the earlier subsection can be used to estimate the Hamming distance in a similar manner.

Algorithm 2 BinSketch estimation of $\mathrm{Ham}(a,b)$

Input: Sketches a_s of a and b_s of b

1: Calculate n_a, n_b, $n_{a,b}$ as done in Algorithm 1
2: **return** approx. of $\mathrm{Ham}(a,b)$ as $ham_{a,b} = n_a + n_b - n_{a,b}$

C. Jaccard similarity

The Jaccard similarity between a pair of binary vectors a and b can be computed from their Hamming distance and their inner product.

$$\mathrm{JS}(a,b) = \frac{\mathrm{IP}(a,b)}{\mathrm{Ham}(a,b) + \mathrm{IP}(a,b)}$$

This paves way for an algorithm to compute Jaccard similarity from BinSketch.

Algorithm 3 BinSketch estimation of $\mathrm{JS}(a, b)$

Input: Sketches a_s of a and b_s of b

Calculate $n_{a,b}$ using Algorithm 1

Calculate $ham_{a,b}$ using Algorithm 2

return approx. of $\mathrm{JS}(a, b)$ as $JS_{a,b} = \dfrac{n_{a,b}}{n_{a,b} + ham_{a,b}}$

D. Cosine similarity

The cosine similarity between a pair binary vectors a and b is defined as:

$$\mathrm{Cos}(a, b) = \mathrm{IP}(a, b) \Big/ \sqrt{|a| \cdot |b|}$$

An algorithm for estimating cosine similarity from binary sketches is straight forward to design at this point.

Algorithm 4 BinSketch estimation of $\mathrm{Cos}(a, b)$

Input: Sketches a_s of a and b_s of b

Calculate n_a, n_b, $n_{a,b}$ as done in Algorithm 1

return approx. of $\mathrm{Cos}(a, b)$ as $cos_{a,b} = n_{a,b} \Big/ \sqrt{n_a \cdot n_b}$

It should be possible to prove that Algorithms 2, 3 and 4 are accurate and low-error estimations of Hamming distance, Jaccard similarity and cosine similarity, respectively; however, those analysis are left out of this paper.

IV. Experiments

a) Hardware description.: We performed our experiments on a machine having the following configuration: CPU: Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz x 4; Memory: 7.5 GB; OS: Ubuntu 18.04; Model: Lenovo Thinkpad T430.

To reduce the effect of randomness, we repeated each experiment several times and took the average. Our implementations did not employ any special optimization.

Datasets.: The experiments were performed on publicly available datasets - namely, 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 [21]; and BBC News Datasets (number of points = 2225, dimension = 9635) [15]. We considered the entire corpus of KOS and BBC News datasets, while for NYTimes, ENRON datasets we sampled 5000 data points.

b) Competing Algorithms: : For our experiments we have used three similarity measures: Jaccard Similarity, Cosine Similarity, and Inner Product. For the Jaccard Similarity, MinHash [5], Densified One Permutation Hashing (DOPH) – a faster variant of MinHash – [27], BCS [26], and OddSketch [23] were the competing algorithms. OddSketch is two-step in nature, which takes the sketch obtained by running MinHash on the original data as input, and outputs binary sketch which maintains an estimate of the original Jaccard similarity. As suggested by authors, we use the number of

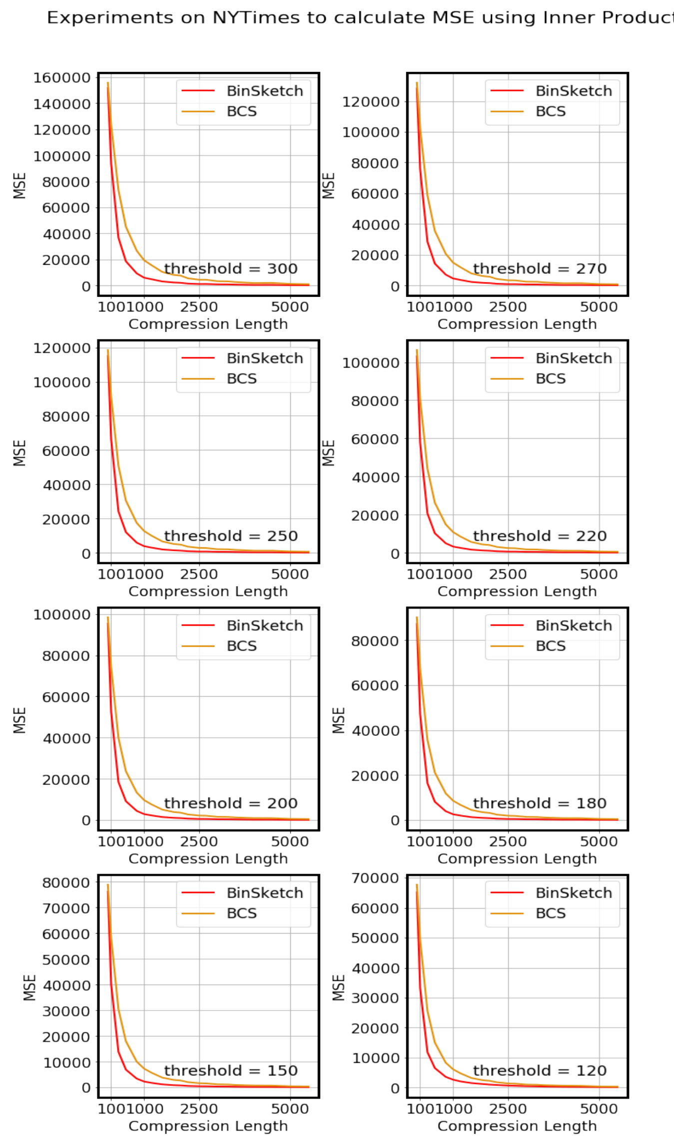

Fig. 1. Comparison of MSE measure on NYTimes datasets. A lower value is an indication of better performance.

MinHash permutations $k = \mathrm{N}/(4(1 - J))$, where J is the similarity threshold. For the Cosine Similarity, SimHash [11], Circulant Binary Embedding (CBE) – a faster variant of SimHash – [31], MinHash [28], and DOPH [27] on the sketch obtained by MinHash [28], were the competing algorithms. For the Inner Product, BCS [26], Asymmetric MinHash [29], and Asymmetric DOPH (DOPH [27] on the sketch obtained by [29]), were the competing algorithms.

A. Experiment 1: Accuracy of Estimation

In this task, we evaluate the *fidelity of the estimate* of BinSketch on various similarity regimes.

a) Evaluation Metric.: To understand the behavior of BinSketch on various similarity regimes, we extract similar pairs – pair of data objects whose similarity is higher than certain threshold –from the datasets. We used Cosine, Jaccard, and Inner Product as our measures. For example: for Jaccard/Cosine case for the threshold value 0.95, we considered only those pairs whose similarities are higher than 0.95. We used mean square error (MSE) as our evaluation criteria. Using $\mathrm{BinSketch}$ and other candidate algorithms, we compressed the datasets to various values of compression length N. We then calculated the MSE for all the algorithms, for different values of N. For example, in order to calculate the MSE of $\mathrm{BinSketch}$ with respect to the ground truth result, for every pair of data points, we calculated the square of the difference between their estimated similarities after the result of $\mathrm{BinSketch}$, and the corresponding ground truth similarity. We added these values for all such pairs and calculated its mean. For Inner Product, we used this absolute value, and for Jaccard/Cosine similarity we computed its negative logarithm base e. A smaller MSE corresponds to a larger $-\log(\mathrm{MSE})$, therefore, a higher value $-\log(\mathrm{MSE})$ is an indication of better performance.

b) Insights.: We summarize our results in Figures 2, and 1 for Cosine/Jaccard Similarity and Inner Product, respectively. For Cosine Similarity, $\mathrm{BinSketch}$ consistently remain to be better than the other candidates. While for Jaccard Similarity, it significantly outperformed *w.r.t.* BCS, DOPH and $\mathrm{OddSketch}$, while its performance was comparable *w.r.t.* $\mathrm{MinHash}$. Moreover, for Inner product 1 results, $\mathrm{BinSketch}$ significantly outperformed *w.r.t.* BCS. We observed a similar pattern on the other datasets as well.

B. Experiment 2: Ranking

Evaluation Metric.: In this experiment, given a dataset and a set of query points, the aim is to find all the points that are similar to the query points, under the given similarity measure. To do so, we randomly, partition the dataset into two parts – 90% and 10%. The bigger partition is called as the *training partition*, while the smaller one is called as *querying partition*. We call each vector of the querying partition as a query vector. For each query vector, we compute the points in the training partition whose similarities are higher than a certain threshold. For Cosine and Jaccard Similarity, we used the threshold values from the set $\{0.95, 0.9, 0.85, 0.8, 0.6, 0.5, 0.2, 0.1\}$. For Inner Product, we first found out the maximum existing Inner product in the dataset, and then set the thresholds accordingly. For every query point, we first find all the similar points in the uncompressed dataset, which we call as ground truth result. We then compress the dataset, using the candidate algorithms, on various values of compression lengths. To evaluate the performance of the competing algorithms, we used the *accuracy-precision-recall* ratio as our standard measure. If the set $\mathcal{O}$ denotes the ground truth result (result on the uncompressed dataset), and the set $\mathcal{O}'$ denotes the results on the compressed datasets, then accuracy = $|\mathcal{O} \cap \mathcal{O}'|/|\mathcal{O} \cup \mathcal{O}'|$, precision = $|\mathcal{O} \cap \mathcal{O}'|/|\mathcal{O}'|$ and recall = $|\mathcal{O} \cap \mathcal{O}'|/|\mathcal{O}|$.

Insights.: We summarize Accuracy results in Figure 4. For Jaccard Similarity, $\mathrm{BinSketch}$ significantly outperformed BCS, DOPH, and $\mathrm{OddSketch}$ while its performance was comparable *w.r.t.* $\mathrm{MinHash}$. For cosine similarity, on higher and intermediate threshold values, $\mathrm{BinSketch}$ outperformed all the other candidate algorithms. However, on the lower threshold values, $\mathrm{MinHash}$ offered the most accurate sketch followed by $\mathrm{BinSketch}$. We observed a similar pattern on the other datasets as well.

a) Efficiency of $\mathrm{BinSketch}$.*:* We comment on the efficiency of $\mathrm{BinSketch}$ with the other competing algorithms and summarize our results in Figure 3. We noted the time required to compress the original dataset using all the competing algorithms. For a given compression length, the compression time of $\mathrm{OddSketch}$ varies based on the similarity threshold. Therefore, we consider taking their average. We notice that the time required by $\mathrm{BinSketch}$ and BCS is negligible for all values of N and on all the datasets. Compression time of CBE is very higher than ours, however, it is independent of the compression length N. After excluding some initial compression lengths, the compression time of $\mathrm{OddSketch}$ is the highest, and grows linearly with N, as it requires running $\mathrm{MinHash}$ on the original dataset. For the remaining algorithms, their respective compression time grows linearly with N.

V. Summary and open questions

In this work, we proposed a simple dimensionality reduction algorithm – $\mathrm{BinSketch}$ – for sparse binary data. $\mathrm{BinSketch}$ offer an efficient dimensionality reduction/sketching algorithm, which compresses a given d-dimensional binary dataset to a relatively smaller N-dimensional binary sketch, while simultaneously maintaining estimates for multiple similarity measures such as Jaccard Similarity, Cosine Similarity, Inner Product, and Hamming Distance, on the same sketch. The performance of $\mathrm{BinSketch}$ was significantly better than BCS [25], [26] while the compression (dimensionality reduction) time of these two algorithms were somewhat very comparable. $\mathrm{BinSketch}$ obtained a significant speedup in compression time *w.r.t* other candidate algorithms ($\mathrm{MinHash}$ [5], [28], $\mathrm{SimHash}$ [11], DOPH [27], CBE [31]) while it simultaneously offered a comparable performance guarantee.

We want to highlight the error bound presented in Theorem 1 is due to a worst-case analysis, which potentially can be tightened. We state this as an open question of the paper. Our experiments on real datasets establish this. For example, for the inner product (see Figure 1), we show that the Mean Square Error (MSE) is almost zero even for compressed dimensions that are much lesser than the bounds stated in the Theorem. Another important open question is to derive a lower bound on the size of a sketch that is required for efficiently and accurately derive similarity values from compressed sketches? Given the simplicity of our method, we hope that it will get adopted in practice.

References

[1] Alekh Agarwal, Oliveier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. volume 15, pages 1111–1133, 2014.

[2] 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.

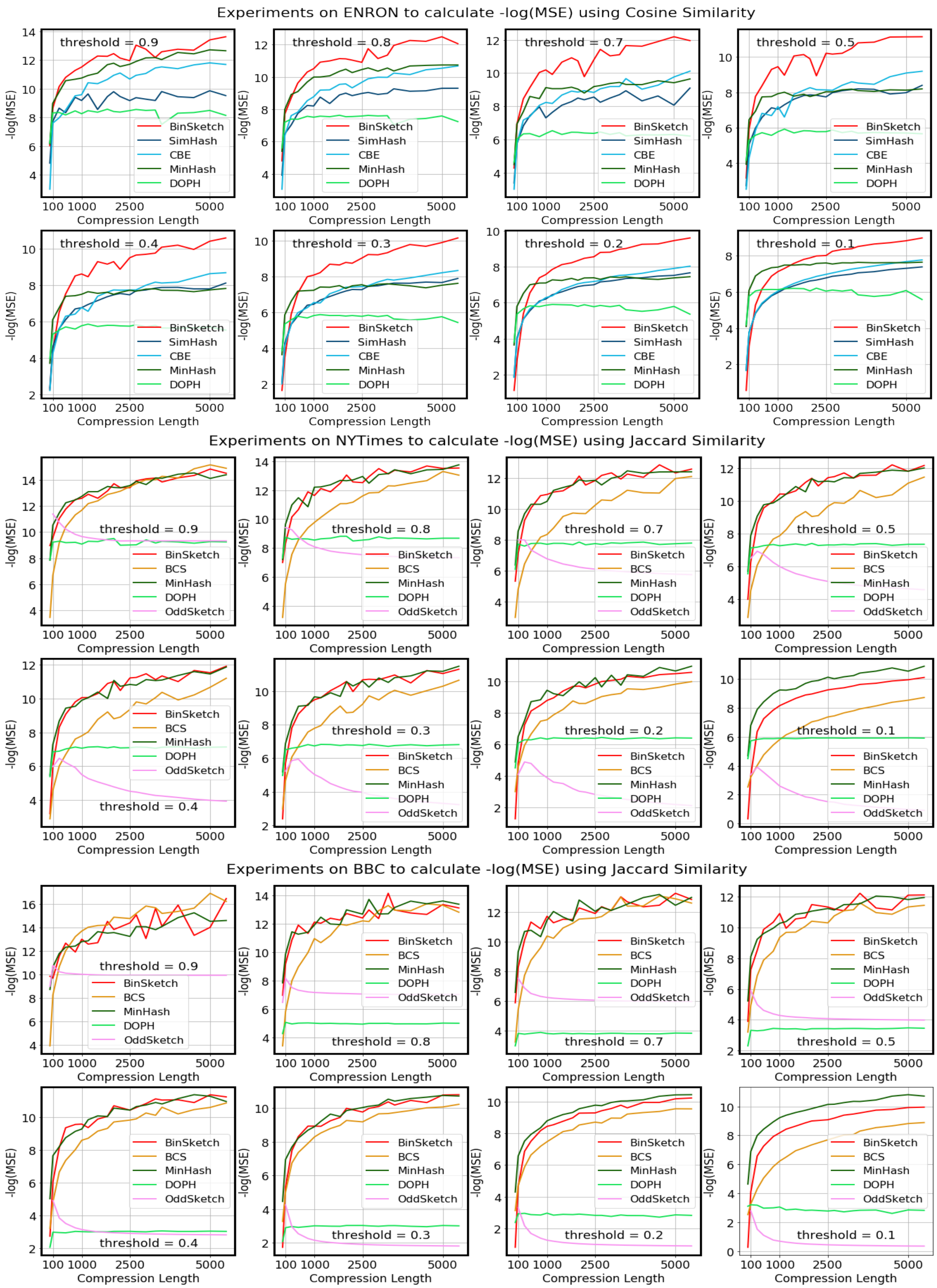

Fig. 2. Comparison of $-\log(\mathrm{MSE})$ measure on Enron, NYTimes, and BBC datasets. A higher value is an indication of better performance.

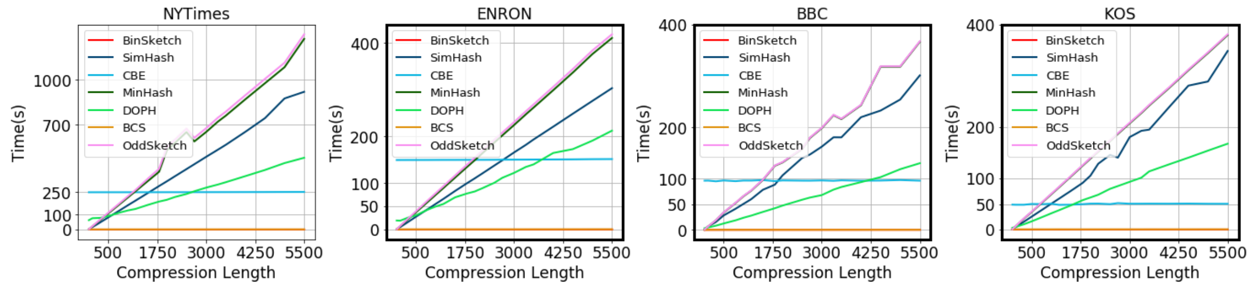

Fig. 3. Comparison of compression times on NYTimes, ENRON, KOS and BBC datasets.

[3] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[4] Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching, 11th Annual Symposium, CPM 2000, Montreal, Canada, June 21-23, 2000, Proceedings*, pages 1–10, 2000.

[5] 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.

[6] Andrei Z. Broder and Michael Mitzenmacher. Survey: Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2003.

[7] 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.

[8] Venkatesan T. Chakaravarthy, Vinayaka Pandit, and Yogish Sabharwal. Analysis of sampling techniques for association rule mining. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 276–283, 2009.

[9] Tushar Chandra, Eugene Ie, Kenneth Goldman, Tomas Lloret Llinares, Jim McFadden, Fernando Pereira, Joshua Redstone, Tal Shaked, and Yoram Singer. Sibyl: a system for large scale machine learning. *Technical report*.

[10] Olivier Chapelle, Patrick Haffner, and Vladimir Vapnik. Support vector machines for histogram-based image classification. *IEEE Trans. Neural Networks*, 10(5):1055–1064, 1999.

[11] 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.

[12] 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.

[13] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001.

[14] 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.

[15] 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.

[16] Matthias Hein and Olivier Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, 2005.

[17] Monika Rauch Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 284–291, 2006.

[18] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, Sep 1998.

[19] 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.

[20] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval, CIVR 2007, Amsterdam, The Netherlands, July 9-11, 2007*, pages 494–501, 2007.

[21] M. Lichman. UCI machine learning repository, 2013.

[22] 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.

[23] 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.

[24] 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, 2018.

[25] 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, 2018.

[26] 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, 2018.

[27] Anshumali Shrivastava. Optimal densification for fast and accurate minwise hashing. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3154–3163, 2017.

[28] Anshumali Shrivastava and Ping Li. In defense of minhash over simhash. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, pages 886–894, 2014.

[29] Anshumali Shrivastava and Ping Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 981–991, 2015.

[30] 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.

[31] Felix X. Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. Circulant binary embedding. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–946–II–954. JMLR.org, 2014.

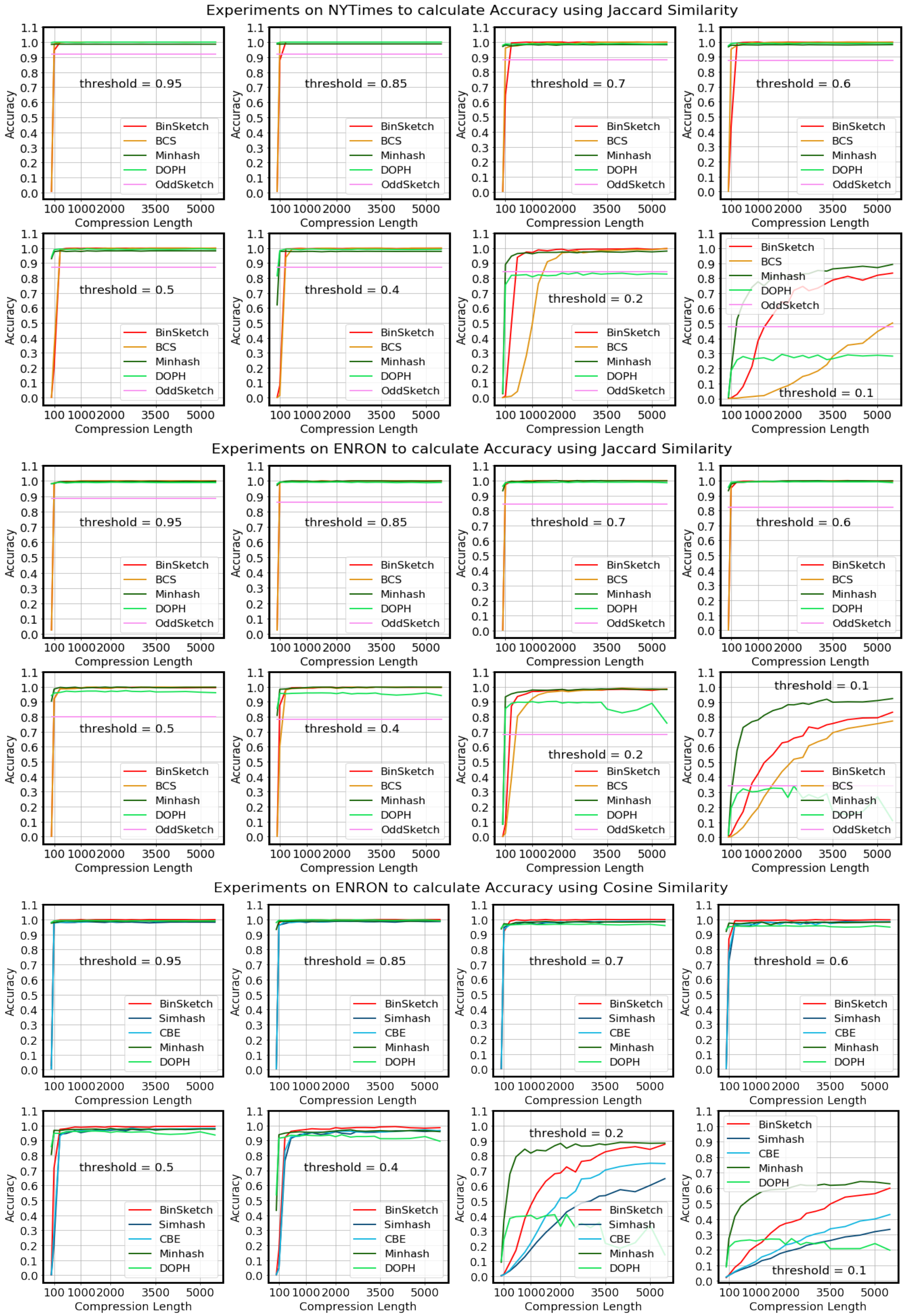

Fig. 4. Comparison of Accuracy measure on NYTimes and ENRON datasets.

APPENDIX A
PROOF OF OBSERVATION 11

In this section we prove that $n^{n_a} \geq 1/2$. For this first we derive an upper bound of $\frac{1}{2}$ on n_{a_s}/N.

Let P denote the expression $\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$ appearing in Lemma 6. Using this lemma, $n_{a_s} \leq \mathbb{E}(|a_s|) + P$. First note that $\frac{P}{N} = \frac{1}{\psi}$ which is at most $\frac{1}{4}$ for $\psi \geq 4$.

Next observe that $\mathbb{E}(|a_s|)/N = (1 - n^{|a|}) \leq (1 - n^{\psi})$ since $|a| \leq \psi$ and $n \in (0,1)$. Furthermore $n^{\psi} = (1 - \frac{1}{N})^{\psi} \geq 1 - \frac{\psi}{N} = 1 - \frac{1}{\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}} \geq \frac{3}{4}$ for practical values of δ and ψ.

Thus we get the upper bound $n_{a_s}/N \leq \frac{1}{N}\left(\mathbb{E}(|a_s|) + P\right) \leq \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.

Algorithm 1 sets $n^{n_a} = 1 - n_{a_s}/N$ which leads us to our required bound that $n^{n_a} \geq \frac{1}{2}$.

APPENDIX B
PROOF OF LEMMA 12

In this section we derive an upper bound on

$$B = \Bigg| |a| - n_a \;+\; |b| - n_b \;+ \frac{1}{\ln\frac{1}{n}} \ln\left[n^{|a|} + n^{|b|} + \frac{\mathbb{E}[\langle a_s, b_s\rangle]}{\mathrm{N}} - 1\right] \;- \frac{1}{\ln\frac{1}{n}} \ln\left[n^{n_a} + n^{n_b} + \frac{n_{a_s,b_s}}{\mathrm{N}} - 1\right] \Bigg|$$

Proof. We first apply triangle inequality and Lemma 8 to obtain

$$B \leq \frac{4/\psi}{\ln\frac{1}{n}} + \frac{4/\psi}{\ln\frac{1}{n}} + \frac{1}{\ln\frac{1}{n}} \left| \ln \frac{n^{n_a} + n^{n_b} + \frac{n_{a_s,b_s}}{\mathrm{N}} - 1}{n^{|a|} + n^{|b|} + \frac{\mathbb{E}[\langle a_s, b_s\rangle]}{\mathrm{N}} - 1} \right|$$

Next we derive an upper bound for the last term. Let U denote $n^{n_a}+n^{n_b}+\frac{n_{a_s,b_s}}{\mathrm{N}}-1$, V denote $n^{|a|}+n^{|b|}+\frac{\mathbb{E}[\langle a_s,b_s\rangle]}{\mathrm{N}}-1$, and W denote $|\ln\frac{U}{V}|$.

By using $n^{n_a} = (1 - \frac{n_{a_s}}{N})$ (and a similar identity for n^{n_b}) as set by Algorithm 1 we obtain that $U = n^{n_a}+n^{n_b}+\frac{n_{a_s,b_s}}{\mathrm{N}}-1 = 1 - \frac{|a_s|+|b_s|-\langle a_s,b_s\rangle}{N}$. Observe that $\langle a_s, b_s\rangle$ is the number of common ones in a_s and b_s and therefore, $|a_s| + |b_s| - \langle a_s, b_s\rangle$ denotes the number of indices at which at least one of a_s *or* b_s is one. This number being at most N, we get that $U \geq 0$ [3].

Using Lemma 5, $n^{|a|}+n^{|b|}+\frac{\mathbb{E}[\langle a_s,b_s\rangle]}{\mathrm{N}}-1 = n^{|a|+|b|+\langle a,b\rangle} > 0$ for non-zero a and b.

The last two observations ensure that the terms inside the logarithm are indeed positive.

Moving forward, now we use triangle inequality to compute

$$|U - V| \leq |n^{n_a} - n^{|a|}| + |n^{n_b} - n^{|b|}| + \frac{|\mathbb{E}[\langle a_s, b_s\rangle] - n_{a_s,b_s}|}{N}$$
$$\leq 3\frac{1}{\psi}(\text{ using Lemma 7 and the next observation})$$

[3]We ignore the $U = 0$ case in good faith since that will happen with extremely low probability; however, this could have been easily tackled by using additional bits in the sketch that are never set to one.

Observation 13. *These claims appear in the proof of Lemma 8:* $|n^{|a|} - n^{n_a}| < \frac{1}{\psi}$ *and* $n^{n_a} \geq n^{|a|} - \frac{1}{\psi}$. *Similarly,* $|n^{|b|} - n^{n_b}| < \frac{1}{\psi}$ *and* $n^{n_b} \geq n^{|b|} - \frac{1}{\psi}$.

Next we show how to lower bound $\max(U, V)$ by using the following observation.

Observation 14. *Using Lemma 7,* $\frac{n_{a_s,b_s}}{N} \geq \frac{\mathbb{E}(\langle a_s,b_s\rangle)}{N} - \frac{1}{\psi}$.

Based on the last two observations we can compute

$$\begin{aligned} U =& n^{n_a} + n^{n_b} + \frac{n_{a_s,b_s}}{N} - 1 \\ \geq& n^{|a|} + n^{|b|} + \frac{\mathbb{E}(\langle a_s, b_s\rangle)}{N} - \frac{3}{\psi} - 1 \\ =& V - \frac{3}{\psi} = n^{|a|+|b|+\langle a,b\rangle} - \frac{3}{\psi} \end{aligned}$$

Therefore, if $U \geq V$, then $\max(U,V) = U \geq V - \frac{3}{\psi}$ and if $V > U$, then $\max(U,V) = V \geq V - \frac{3}{\psi}$. This leads to:

$$\begin{aligned} \max(U,V) &\geq V - \frac{3}{\psi} = (1 - \frac{1}{N})^{|a|+|b|+\langle a,b\rangle} - \frac{3}{\psi} \\ &\geq 1 - \frac{|a| + |b| + \langle a, b\rangle}{N} - \frac{3}{\psi} \geq 1 - \frac{3\psi}{N} - \frac{3}{\psi} \\ &\geq 1 - \frac{3}{\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}} - \frac{3}{\psi} \end{aligned}$$

which is at least $\frac{1}{2}$ for reasonable values of ψ and δ.

Next we upper bound W by employing the inequality $\left|\ln\frac{A}{B}\right| \leq \frac{|A-B|}{\max(A,B)}$ that holds for non-negative A, B and can be derived from the standard inequality $\ln x \leq x - 1$ for $x > 0$. Here, set $A = U$ and $B = V$ and obtain that $W \leq \frac{3/\psi}{1/2} = \frac{6}{\psi}$.

Now we gather all the upper bounds of the expressions appearing in B and compute the final upper bound.

$$B \leq \frac{4/\psi}{\ln\frac{1}{n}} + \frac{4/\psi}{\ln\frac{1}{n}} + \frac{6/\psi}{\ln\frac{1}{n}} = \frac{14/\psi}{\ln\frac{1}{n}} \leq \frac{14N}{\psi} = 14\sqrt{\frac{\psi}{2}\ln\frac{2}{\delta}}$$

Of course this bound holds when the upper bounds on $(|a| - n_a)$, $(|b| - n_b)$ and $(\mathbb{E}[\langle a_s, b_s\rangle] - n_{a_s,b_s})$ are correct and each of them is incorrect with probability at most δ. Therefore, using Union-bound, we can say that our upper bound as required in the lemma can be incorrect with probability at most 3δ. $\square$

Appendix C
Extended Experimental Results

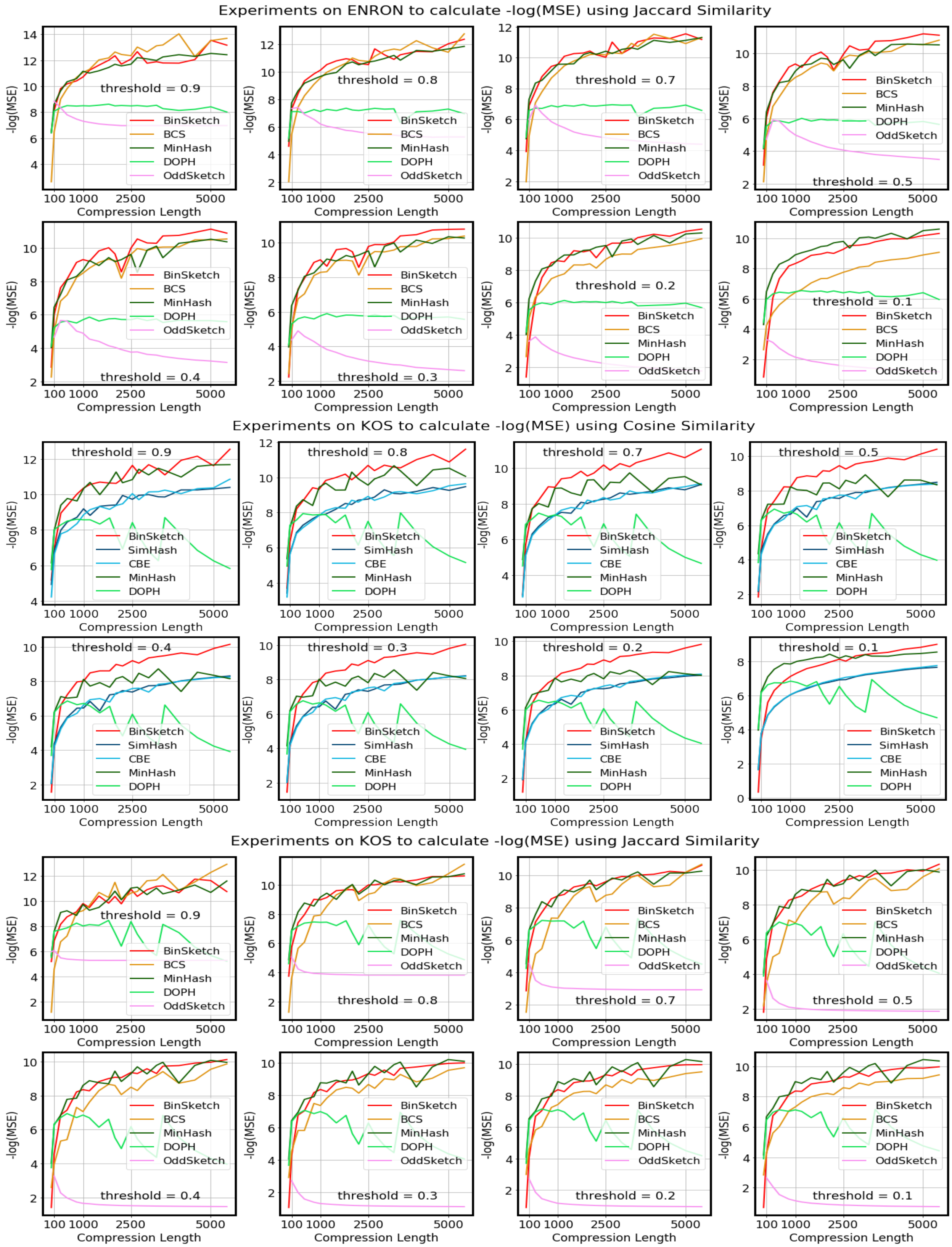

Fig. 5. Comparison of $-\log(\mathrm{MSE})$ measure on ENRON and KOS datasets.

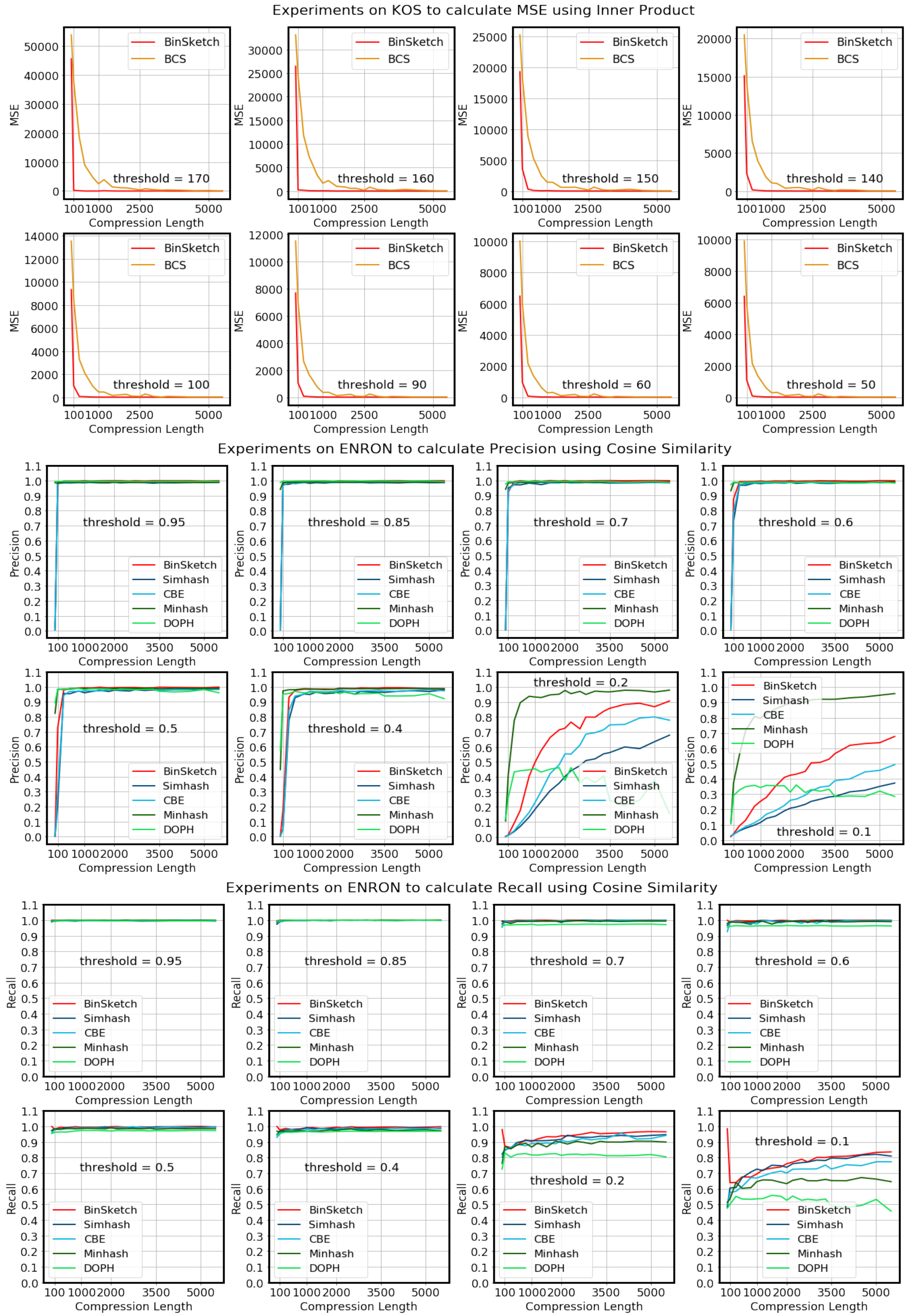

Fig. 6. Comparison of MSE on KOS dataset for Inner Product, and comparison of Precision and Recall on ENRON dataset for Cosine Similarity.

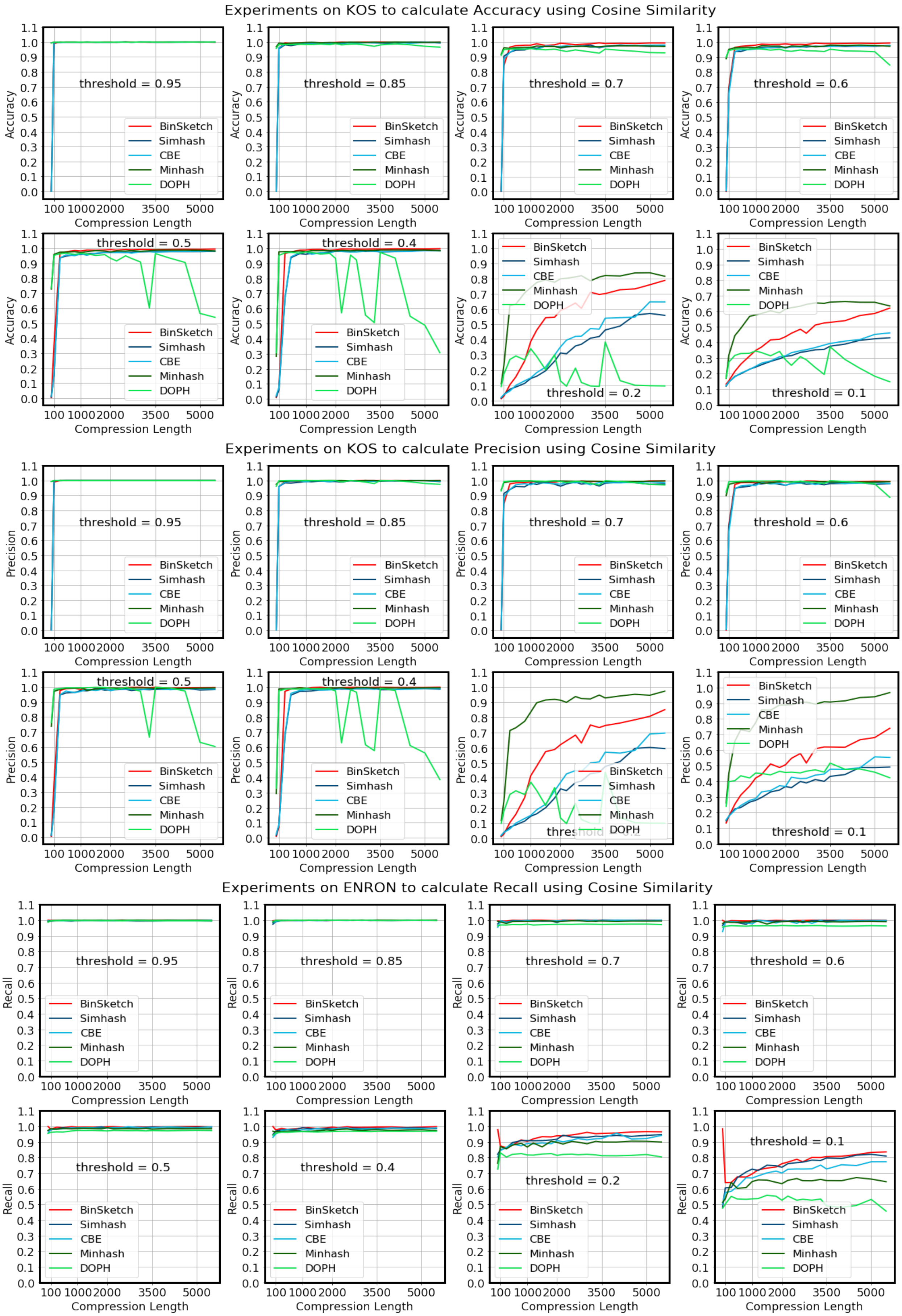

Fig. 7. Comparison of Accuracy, Precision, Recall measure on KOS datasets for Cosine Similarity.

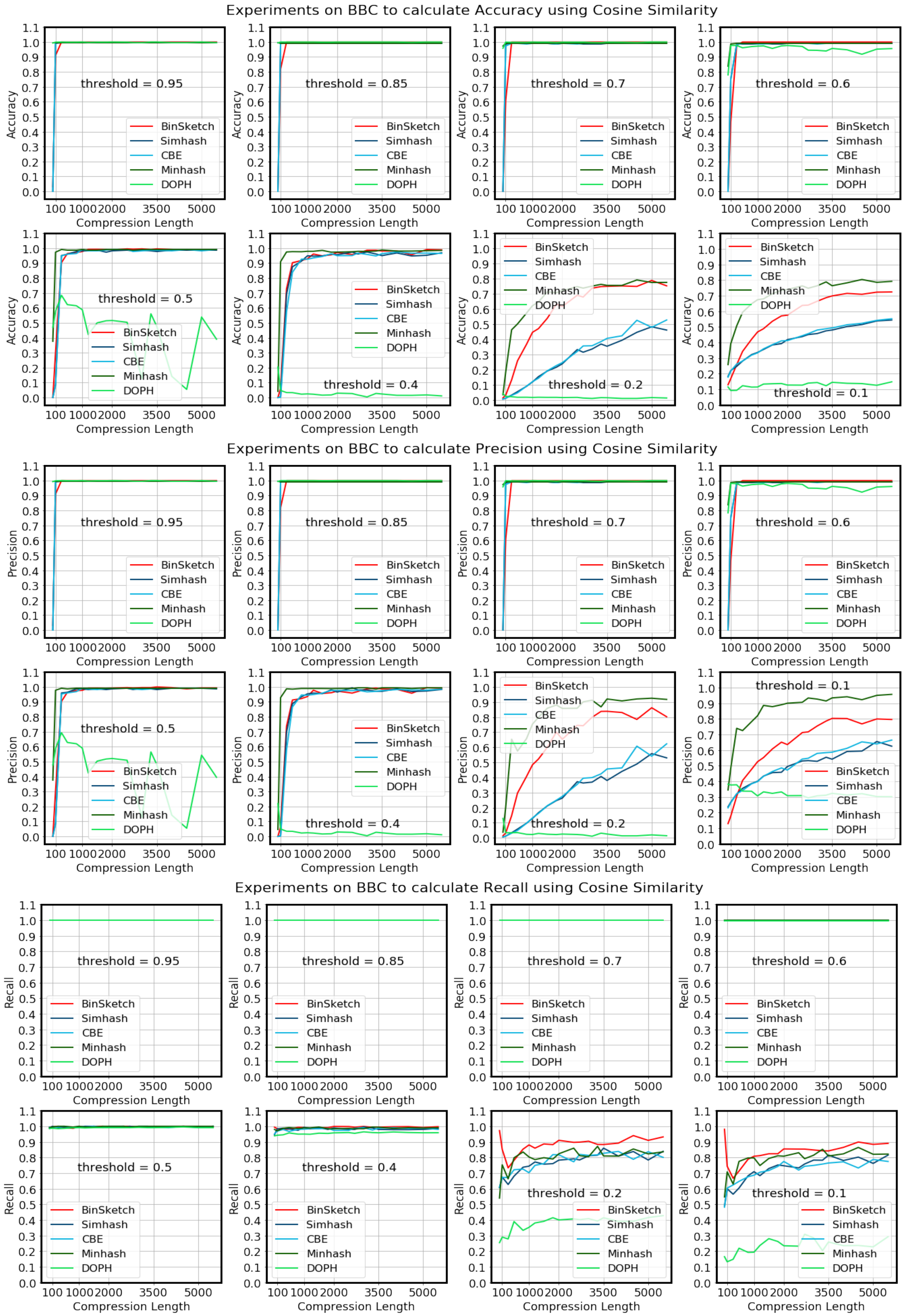

Fig. 8. Comparison of Accuracy, Precision, Recall measure on BBC datasets for Cosine Similarity.

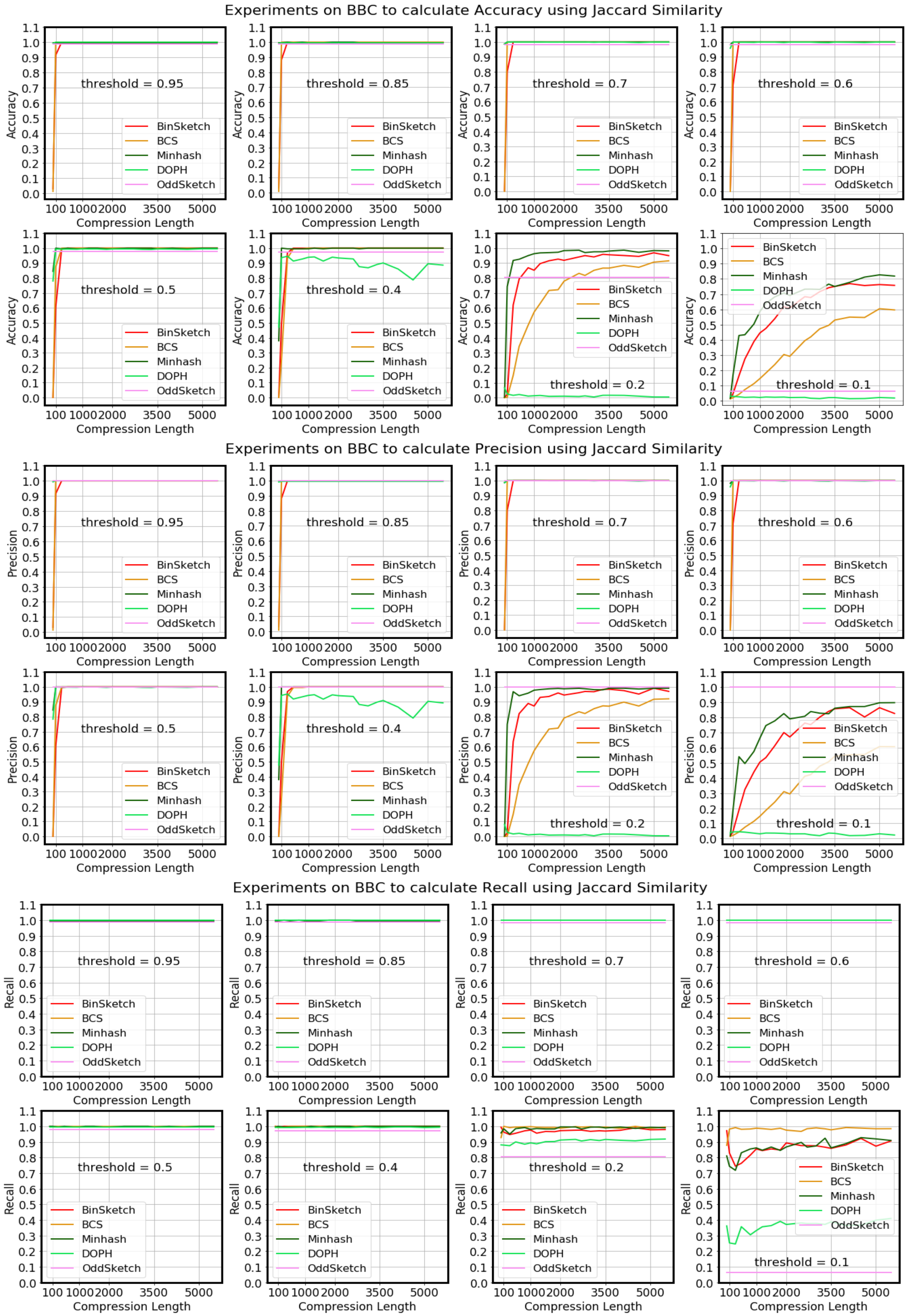

Fig. 9. Comparison of Accuracy, Precision, Recall measure on BBC datasets for Jaccard Similarity.

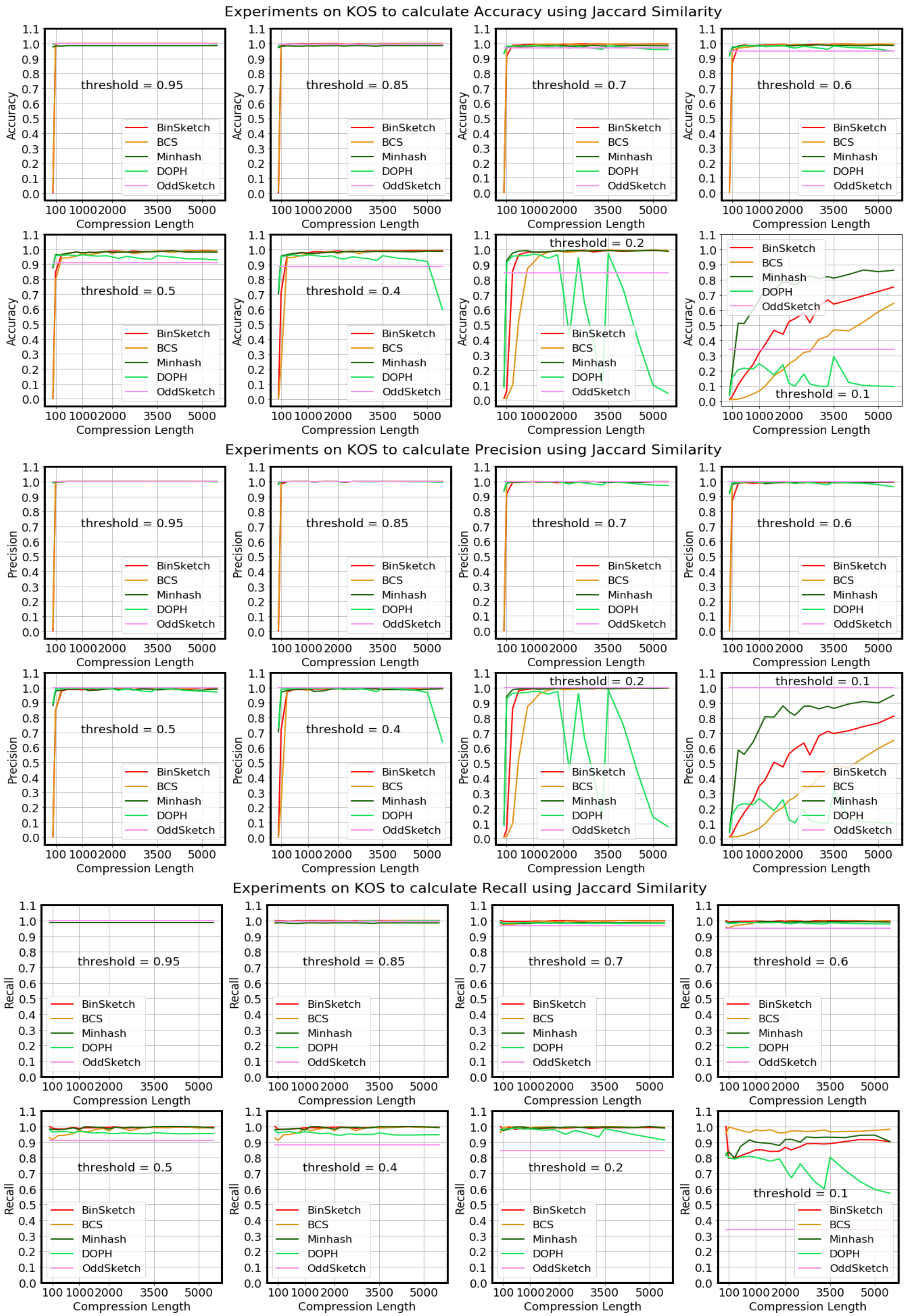

Fig. 10. Comparison of Accuracy, Precision, Recall measure on KOS datasets for Jaccard Similarity.

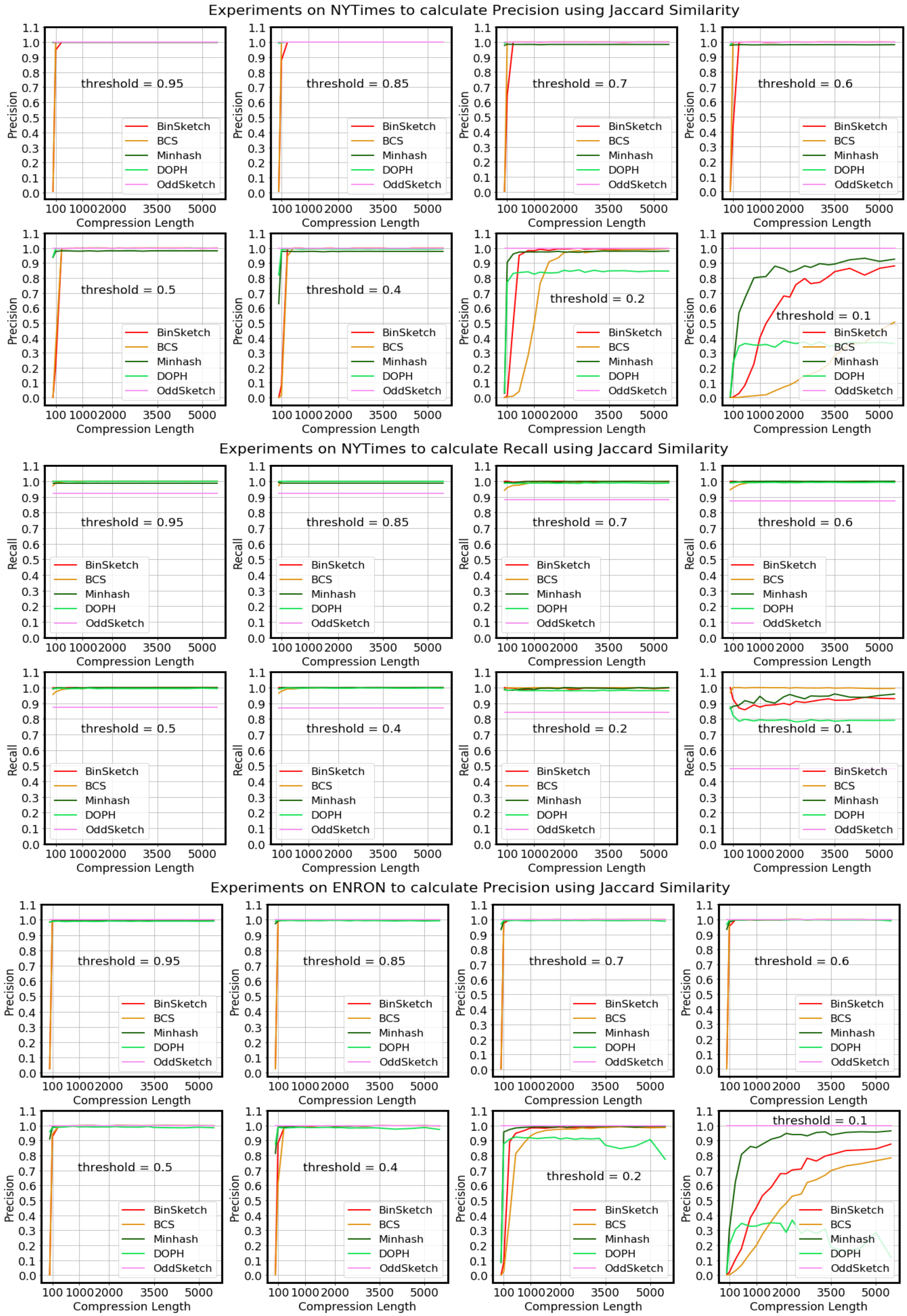

Fig. 11. Comparison of Precision, Recall on NYTimes and Precision on ENRON datasets for Jaccard Similarity.

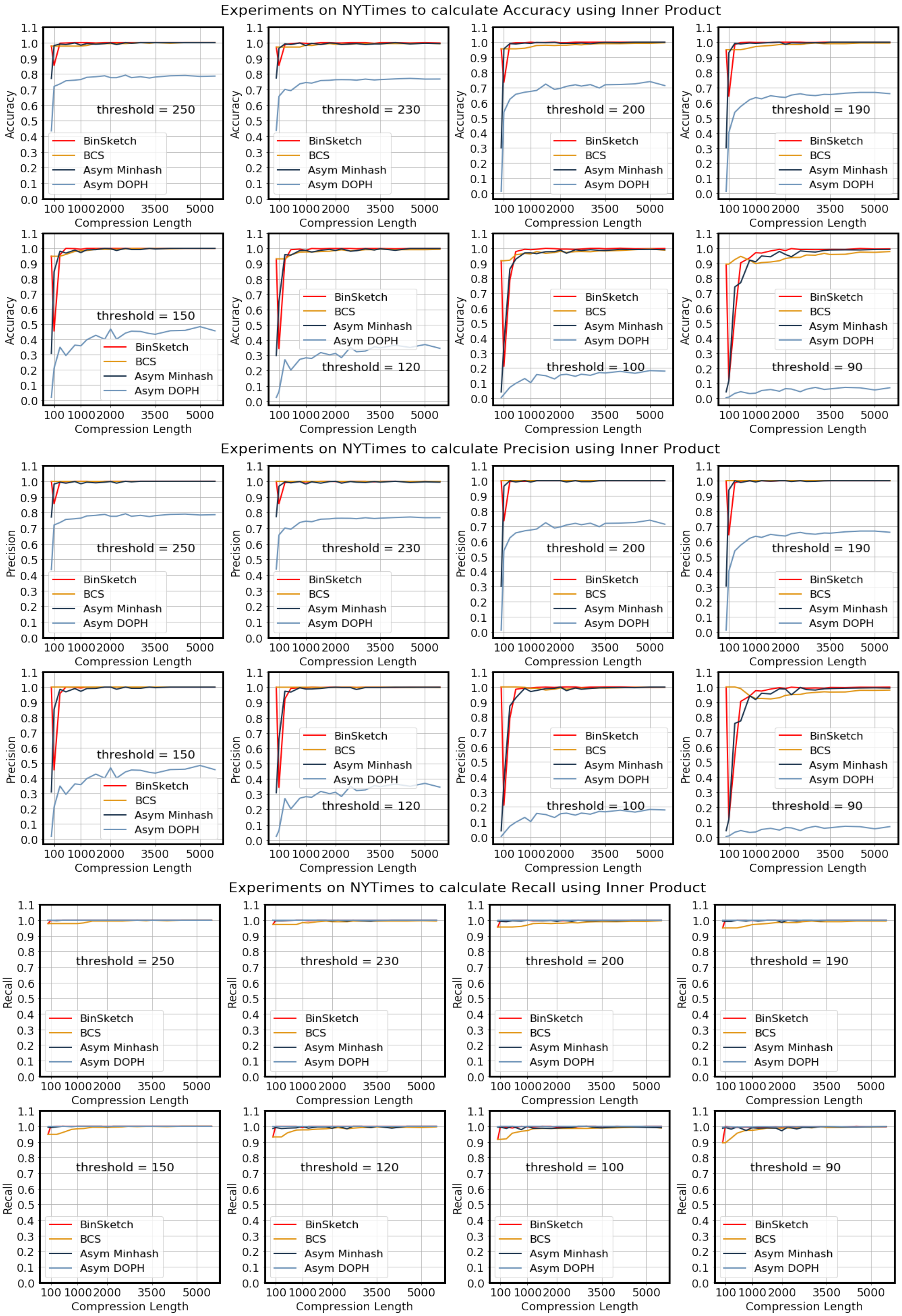

Fig. 12. Comparison of Accuracy, Precision, Recall measure on NYTimes datasets for Inner Product.

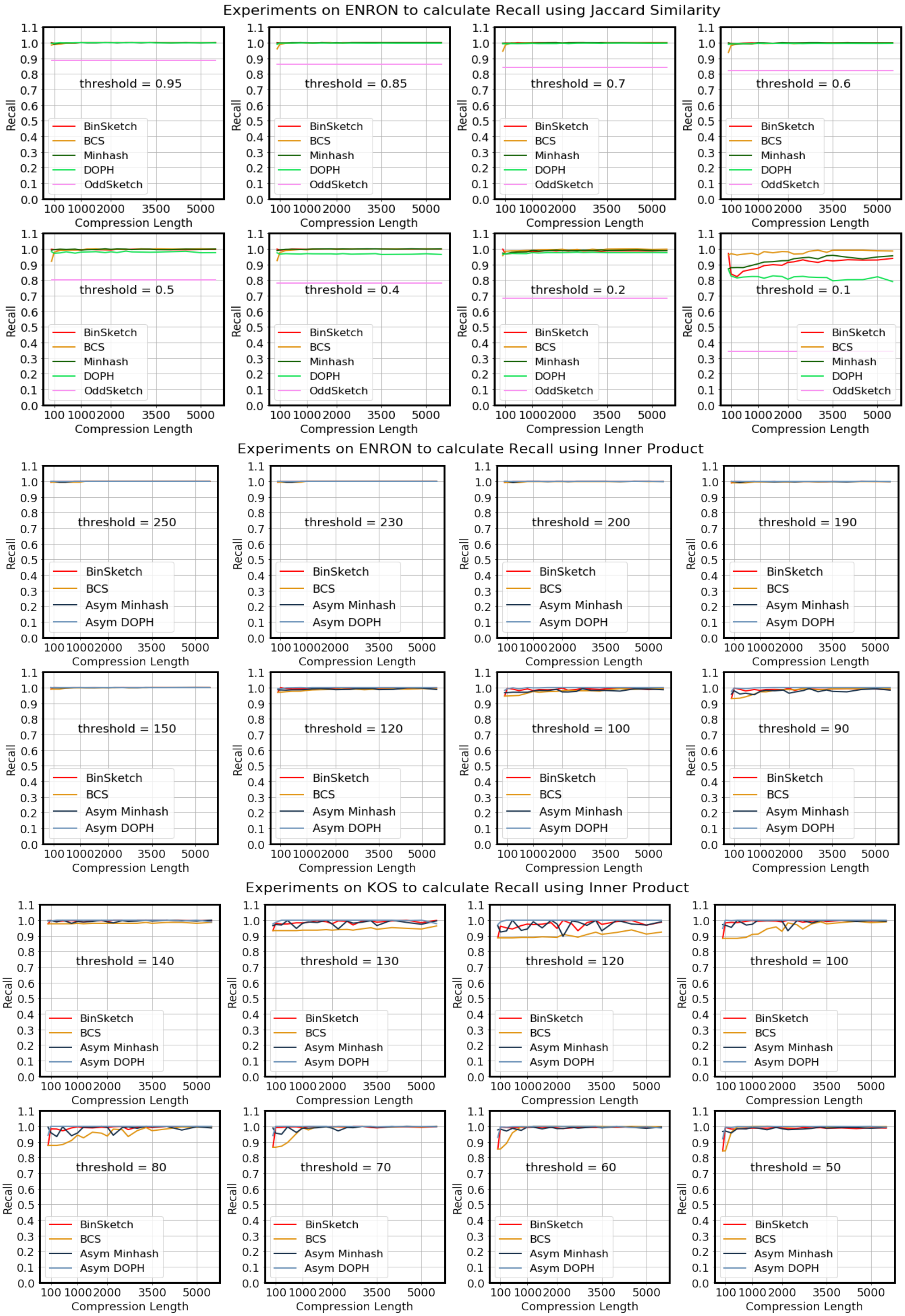

Fig. 13. Comparison of Recall measure on ENRON for Jaccard Similarity and ENRON and KOS for Inner Product.