Tight bounds for membership filters with two-sided errors

Anxin Guo
Computer Science Department, Northwestern University, USA
anxinbguo@gmail.com

October 6, 2025

Abstract

We study two-sided membership filters, a natural generalization of classical Bloom-like filters that allows both false positive and false negative errors. Two-sided filters are practically motivated by a number of applications, and they can potentially use less space by tolerating both types of errors.

We establish the fundamental space lower bound for two-sided filters: any filter achieving false positive rate ε_P and false negative rate ε_N must use at least $D_{KL}(1-\varepsilon_N \| \varepsilon_P)$ bits per key, the binary Kullback-Leibler divergence between $\mathrm{Bern}(1-\varepsilon_N)$ and $\mathrm{Bern}(\varepsilon_P)$. This bound generalizes the classical $\log(1/\varepsilon_P)$ lower bound for one-sided filters and provides a natural and complete characterization of the space-error trade-off.

We also present a simple but inefficient construction that achieves this lower bound, demonstrating its tightness. Additionally, we show that unlike one-sided filters, aggregating multiple two-sided filters can lead to suboptimal space usage, highlighting fundamental differences between the two settings.

1 Introduction

Approximate membership query filters (or simply *filters*), such as Bloom filters [6], have achieved tremendous success in many applications including networking [8], database, distributed system [32], biology [24] and others [23]. These data structures can "store" a set of *keys* S in some universe U, using space linear in $|S|$, regardless of the size of U. Information-theoretically, storing such a set of size n takes $\log\binom{|U|}{n}$[1] bits, which is prohibitively large as filters are typically used when $|U| \gg n$. Hence this compactness, while impressive, must come with some inaccuracies. Typically, this inaccuracy is characterized by a small *false positive rate* (FPR) ε of the user's choice, such that some ε portion of the *non-keys* $U \setminus S$ will be recognized as keys by the filter. Meanwhile, the false negative rate (FNR) is usually guaranteed at zero.

Since Bloom's proposal of the first filter in 1970, various works in data structure design have been devoted to designing filters that are space-optimal, computationally cheap, and feature-rich (allowing for deletion, for instance). Many filters designs since then, including variants of the quotient [3, 2, 28] and cuckoo [15, 13, 7, 33] filters, as well as other structures for storing (multi)set of *fingerprints* of the keys [26, 5, 14, 4], can achieve space usage $O(n)$-close to the $n \log \frac{1}{\varepsilon}$ bits lower bound, shown by Carter et al. [9]. Some other designs can even achieve this lower bound up to $o(n)$ extra bits [11, 29] by considering the closely related *retrieval* data structures (defined in section 2.1).

It seems clear then, that $\log \frac{1}{\varepsilon}$ characterizes the fundamental trade-off between per-element space and error for filters. However, as the term "approximate membership" suggests, all existing filters are special cases of a more general class of data structures: those which store S with *both false negative and false positive*, which potentially use even less space than the classical filters. Despite being a natural generalization motivated by a number of practial applications [19], there has been no theoretical characterization of the trade-off between space, FPR, and FNR for two-sided filters [2]. This leads to the following natural question:

What is the minimum space required for a two-sided filter that stores a set of size n with FPR of ε_P and FNR of ε_N, independent of the universe size?

1.1 Our results

Our first result is a space lower bound for data structures storing n keys in a universe of size u, up to FPR of ε_P and FNR of ε_N, for the case where key density n/u is fixed. This lower bound applies not only to filters (which operate regardless of the universe size), but also for other approximate membership structures tailor made for known universes:

Theorem 1. *For any fixed $p = \frac{n}{u}$, as $n \to \infty$, any data structure storing key set of size n with FPR of $\varepsilon_P \in (0,1)$ and FNR of $\varepsilon_N \in (0, 1-\varepsilon_P)$ must use at least*

$$D_{KL}(1-\varepsilon_N \| \varepsilon_P) - \frac{(1-\varepsilon_N-\varepsilon_P)^2}{2\varepsilon_P(1-\varepsilon_P)\ln 2} \cdot p + O(p^2)$$

bits of space per key. Here, $D_{KL}(q_1\|q_2)$ denotes the binary Kullback-Leibler divergence of a Bern(q_1) *distribution from a* Bern(q_2) *distribution:*

$$D_{KL}(q_1\|q_2) = q_1 \log \frac{q_1}{q_2} + (1-q_1)\log\frac{1-q_1}{1-q_2}.$$

For two-sided filters which are *universe-independent*, by considering the usual setting where $u \gg n$, we immediately obtain the following corollary:

[1]All logarithms are base-2 in this paper.

[2]From now on, we use *two-sided* filters to denote such structures with two-sided errors, and the classical filters will be referred to as *one-sided* filters.

Corollary 2. *Any two-sided filter storing n keys with FPR of $\varepsilon_P \in (0,1)$ and FNR of $\varepsilon_N \in (0, 1-\varepsilon_P)$ must use at least $D_{KL}(1-\varepsilon_N \| \varepsilon_P)$ bits of space per key.*

This lower bound generalizes the one-sided lower bound $\log \frac{1}{\varepsilon_P}$, as $D_{KL}(1\|\varepsilon_P) = \log \frac{1}{\varepsilon_P}$ for any $\varepsilon_P \in (0,1)$. Moreover, this value is intuitive from a statistical perspective: by drawing n samples from a large population with positive rate ε_P, the probability of obtaining positive rate at least $1-\varepsilon_N$ on the samples is $2^{-n(D_{KL}(1-\varepsilon_N\|\varepsilon_P)+o(n))}$. The "amount of information" needed to describe such a sample is therefore around $n \cdot D_{KL}(1-\varepsilon_N\|\varepsilon_P)$ bits.

The proof for Theorem 1 is based on lossy data compression, formalizing the above intuition and extending the idea of [19]. Some technicality arises from the fact that the membership informations (whether $x \in S$ or not) for each $x \in U$ is dependent on the membership information of the rest of U. To obtain a rigorous lower bound, we apply the law of large numbers and prove a generalized version of the rate-distortion theorem for two error metrics.

It's worth noting that our result also refines an early result [27] for "lossy dictionaries", a combination of two-sided filter and retrieval structure. In particular, for two-sided filters, the authors gave a lower bound of

$$\begin{aligned}&(1-\varepsilon_N)\log\frac{1}{\varepsilon_P+p} - \Theta(n)\\ =&(1-\varepsilon_N)\log\frac{1}{\varepsilon_P} - \Theta(n) - \frac{1-\varepsilon_N}{\varepsilon_P \ln 2}\cdot p + O\left(p^2\right)\end{aligned}$$

bits per element. Theorem 1 characterizes the $\Theta(n)$ term, and refined the dependency on p. Their result was based on counting arguments similar to [9], which could explain the discrepancy.

Finally, to complement the lower bound, we construct a simple two-sided filter that achieves the space lower bound based on lossy dictionaries, thereby demonstrating its tightness:

Theorem 3. *There exists a two-sided filter with space usage $n \cdot D_{KL}(1-\varepsilon_N\|\varepsilon_P) + o(n)$ bits per key, for any $\varepsilon_P \in (0,1)$ and $\varepsilon_N \in (0, 1-\varepsilon_P)$.*

We note that this construction is not practical. In fact, it is unlikely we can construct this filter design in polynomial time due to a reduction to learning with errors (LWE) problem [30]. It is open whether we can construct such a filter in polynomial time, or if there is some inherent hardness in the problem.

1.2 Why introduce false negatives?

Despite its success, one-sided filters sometimes have limited performance and versatility due to the stringent no-false-negative requirement. As pointed out by Hurley and Waldvogel [19], certain filter applications in networking [8] would tolerate or even *prefer* false negatives. An example is *set difference reconciliation*, where Bob sends a filter of his local storage S_B to Alice, such that Alice can compute $S_A \setminus S_B$ and update Bob on what he does not have. Here, a false negative in Bob's filter only causes a minor overhead, yet a false positive will lead to synchronization errors. For more examples, see table 1.

Moreover, even in traditionally FPR-heavy tasks, the idea of introducing FNR to further reduce FPR is oftentimes employed, as exemplified by designs such as retouched Bloom filter [12], generalized Bloom filter [21], and autoscaling Bloom filter [20]. In *the Bloom paradox* [31], the authors also discussed *selective insertion* and *selective query* as a means to reduce FPR and introduce FNR, in order to minimize the overall cost. All these approaches can be viewed as heuristics for the FPR-FNR trade-off for a given space usage, and it is therefore of great interest to study the fundamental limit of this approach, as well as methods that can achieve this limit.

The connections to retrieval and lossy dictionary structures further motivate our investigation into two-sided filters, as these related fields have demonstrated the value of allowing controlled information loss for improved space efficiency and performance. Understanding the fundamental

Application area	Necessity of one-sided error
Distributed caching	No
Object location in P2P systems	No
Approximate set reconciliation	False negative preferred
Resource routing	No
Loop detection	False negative preferred
Flow detection	Yes
Multicast	Yes
Hyphenation Exceptions	Yes
Set intersection	Yes
Differential files	Yes

Table 1: A summary of network application of filters with the role of false negatives highlighted. This table is taken from Hurley and Waldvogel [19].

limits of two-sided filters thus contributes not only to filter design but also to the broader understanding of space-efficient data structures with approximate semantics.

We discuss other related works in section 2.4.

2 Preliminaries

2.1 Approximate Membership, Filters, and Retrieval

In this paper, we use the generic term "approximate membership structure" to denote any data structure that stores a set of keys with both false positive and false negative which operates for a fixed, known universe U. One can think of U as the set $[2^w]$, where w is the word length in a RAM model, a standard assumption in the literature [26]. This structure behaves significantly different from the *filters* we define below. We will not focus on this structure, but our methods is inspired by the study of such structures [19].

Definition 4 (Approximate Membership Structure)**.** A ***approximate membership structure*** $\mathcal{M}$ is a probabilistic data structure that stores a set of keys with both false positive and false negative errors. $\mathcal{M}$ implements the following two operations:

1. On input ε_P, ε_N, key set S, and universe U, $\mathcal{M}$ is initialized by storing an internal state of some $L_{\mathcal{M}}(|S|, |U|, \varepsilon_P, \varepsilon_N)$ bits.

2. On query $x \in U$, (by an abuse of notation) $\mathcal{M}$ outputs some $\mathcal{M}(x) \in \{0, 1\}$. The query answer should satisfy:
$$\begin{cases} \mathbb{P}_{x\sim\mathrm{Unif}(U\setminus S)}[\mathcal{M}(x) = 1] & \leq \varepsilon_P, \\ \mathbb{P}_{x\sim\mathrm{Unif}(S)}[\mathcal{M}(x) = 0] & \leq \varepsilon_N, \end{cases}$$
where the probabilities are taken over the random oracle functions used in both initialization and query.

Now we define filters. To capture the requirement that filters operate regardless of the universe size, we require the space usage to be independent of the universe size, assuming access to random oracles functions on the universe U.

Definition 5 (Two-Sided Filter)**.** A ***two-sided filter*** $\mathcal{F}$ is a probabilistic data structure with access to binary random oracle functions $f_1, f_2, \ldots : U \to \{0, 1\}$. $\mathcal{F}$ implements the following two operations using these random oracle functions, regardless of U:

1. On input ε_P, ε_N, and key set S, $\mathcal{F}$ is initialized by storing an internal state of some $L_{\mathcal{F}}(|S|, \varepsilon_P, \varepsilon_N)$ bits.

2. On query $x \in U$, (by an abuse of notation) $\mathcal{F}$ outputs some $\mathcal{F}(x) \in \{0,1\}$. The query answer should satisfy:

$$\begin{cases} \mathbb{P}_{x \sim \mathrm{Unif}(U \setminus S)}[\mathcal{F}(x) = 1] & \leq \varepsilon_P, \\ \mathbb{P}_{x \sim \mathrm{Unif}(S)}[\mathcal{F}(x) = 0] & \leq \varepsilon_N, \end{cases}$$

where the probabilities are taken over the random oracle functions used in both initialization and query.

We call $\mathcal{F}$ ***one-sided*** if it has $\varepsilon_N = 0$. We sometimes use "error rates $(\varepsilon_P, \varepsilon_N)$" to denote an FPR of ε_P and FNR of ε_N.

For completeness, we also define the closely related retrieval data structure:

Definition 6 (Retrieval)**.** A ***retrieval*** data structure $\mathcal{R}$ is a compact dictionary-like structure that associates a k-bit value $v_1, \dots, v_n$ with each of the n keys in $\{x_1, \dots, x_n\} = S \subseteq U$. Upon a query $x \in U$,

- If $x = x_i \in S$ is a key, then $\mathcal{R}$ must return the correct value v_i.
- If $x \in U \setminus S$ is a non-key, then $\mathcal{R}$ can return an arbitrary value in $\{0,1\}^k$.

This data structure has space lower bound nk, since it can easily implement a one-sided filter with FPR of 2^{-k} via the following procedure [11][29]:

1. Set the associate value of key $x_i \in S$ to be $v_i = f(x_i)$, for random function $f : U \to \{0,1\}^k$.
2. Construct retrieval structure $\mathcal{R}$ which stores value v_i for each key x_i.
3. When queried a key $x \in U$, the filter queries $\mathcal{R}$ with the same x and obtains value v. Filter returns 1 iff $f(x) = v$.

2.2 Two types of space-optimality

Given the space lower bound, we want to distinguish two types of filters that are often referred to as "optimal" in the literature:

Definition 7. Membership filter $\mathcal{F}$ is ***strongly optimal*** if for any pair of fixed $\varepsilon_P, \varepsilon_N$,

$$\frac{1}{n} L_{\mathcal{F}}(n, \varepsilon_P, \varepsilon_N) = D_{KL}(1 - \varepsilon_N \| \varepsilon_P) + o(1), \text{ as } n \to \infty.$$

Meanwhile, $\mathcal{F}$ is ***weakly optimal*** if for any $\varepsilon_P, \varepsilon_N$,

$$\frac{1}{n} L_{\mathcal{F}}(n, \varepsilon_P, \varepsilon_N) \leq D_{KL}(1 - \varepsilon_N \| \varepsilon_P) + O(1), \text{ as } n \to \infty.$$

Most practical filters claiming to achieve optimality (see section 2.4.1) are only *weakly* optimal, even as their load approaches 1 and becomes practically infeasible. This is for good reason: any one-sided filter supporting insertion must have an $O(n)$ multiplicative overhead, and thus *cannot be* strongly optimal [22]. Moreover, ε_P is often set to be small in practice, and by taking $\varepsilon_P \to 0$ and ε_N constant, the $D_{KL}(1 - \varepsilon_N \| \varepsilon_P)$ term dominates, and weakly optimal filters will have a negligible multiplicative overhead in this limit. It's when $\varepsilon_P + \varepsilon_N \approx 1$ that the strong optimality becomes significantly better.

In this paper, we focus solely on *strongly* optimal two-sided membership filters, as we are interested in the fundamental trade-off between space, FPR, and FNR. Another reason is the following simple baseline is, in fact, already weakly optimal.

2.3 Selective insertion: a baseline

Selective insertion is a simple two-sided filter proposed by Rottenstreich and Keslassy [31], as a solution to the Bloom paradox. Essentially, we use a one-sided filter with FPR of ε_P to store only $1-\varepsilon_N$ portion of the keys S. This structure clearly achieves the desired error levels, and its space function is:

$$(1-\varepsilon_N)n\log\frac{1}{\varepsilon_P} = n(D_{KL}(1-\varepsilon_N\|\varepsilon_P) + \underbrace{H(\varepsilon_N)+\varepsilon_N\log(1-\varepsilon_P)}_{\text{overhead.}})$$

For $\varepsilon_P+\varepsilon_N<1$, the overhead is between 0 and 1 bit per key, where the value 1 is taken when $\varepsilon_N=\frac{1}{2}$ and $\varepsilon_P\to 0$. We thus conclude that this structure is weakly optimal. Furthermore, this is the best possible for any structure that *fixes* an $1-\varepsilon_N$ portion of keys to keep, so a strongly optimal filter must randomize on which exact keys to store.

However, this baseline has *unbounded* multiplicative overhead when $\varepsilon_P+\varepsilon_N\approx 1$: the lower bound approaches zero at this limit, yet the selective insertion method still requires $\Omega(1-\varepsilon_N)$ space per key, even when $\varepsilon_P\to 1-\varepsilon_N$.

2.4 Related works

In this subsection, we focus mainly on space-optimal (one-sided) membership filters in the literature. We also consider several closely related problems, to compare their space lower bound techniques with ours. Some modern designs choose to give up (even weak) space optimality for simplicity and practical performance [17][18], but they are less relevant for our purpose.

2.4.1 Hashing-based fingerprint storage

The gist of many contemporary filters is to store the *fingerprints* $\{f(x_1),\ldots,f(x_n)\}$ of the keys $S=\{x_1,\ldots,x_n\}$ in a hash table-like structure. Here, $f:U\to\{0,1\}^k$ is a random (hash) function, and each k-bit string $f(x_i)$ is stored one of the n entries of a table. Suppose we have perfect hash function $h:U\to[n]$ that is bijective on S, then we can simply store the string $f(x_i)$ in the $h(x_i)$th entry, and since we don't have to resolve hash collisions, this data structure achieves FPR of 2^{-k} with exactly nk bits.

The drawback of this idea is that a perfect function takes $\geq n\log e$ bits to store [1]. In a sense, this $\Omega(n)$ overhead is intrinsic to the hash collision issue, and it is present regardless of the methods for resolving such collisions. Consequently, all variants of the Cuckoo filter [15][13][7][33], quotient filter [3][2][28], and other structures for storing (multi)set of fingerprints [9][26][5][14][4], all suffer from this overhead even near full load, where structures like Cuckoo filter becomes prohibitively slow. Thus, despite their success in various other aspects, these filters cannot possibly be strongly optimal.

2.4.2 The retrieval problem

In contrast, another class of *retrieval-based* filters do not suffer from the same overhead. Instead of hashing, they use retrieval methods to associate keys $x\in S$ with random values $f(x)$, avoiding any possibility for collision. The cost is that S must be *static* set that never changes. The association of filters with retrieval is proposed independently by Dietzfelbinger and Pagh [11] and Porat [29].

2.4.3 Space lower bounds and related problems

The counting argument. The classical space lower bound of $n\log\frac{1}{\varepsilon}$ is proven in 1978 by Carter et al. [9]. They applied a counting argument, calculating how many sets of size $n+\varepsilon(u-n)$ (the set each filter actually recognizes) is needed, in order to cover all $\binom{u}{n}$ possible sets of size n.

While inspiring, this particular reasoning only applies to filters with a fixed FPR of ε, whereas some important filters can only achieve ε error *in expectation*[3]. In this case, each configuration of the filter could potentially recognize sets of various sizes, depending on the random tape.

Notably, Pagh and Rodler [27] applied the same counting argument to deduce a lower bound for general membership filters: $(1-\varepsilon_N)\log\frac{1}{\varepsilon_P+n/u} - \Theta(n)$ bits. They applied this bound to study *lossy dictionaries*, a two-sided variant of the retrieval problem. Their bound agrees with ours up to $\Theta(n)$, but specifying the exact expression via the counting argument seems far from easy.

Rate distortion argument. The rate distortion method takes into account the randomness in filter construction and query, as well as *expected* error rates. Our techniques are partially inspired by Hurley and Waldvogel [19], who used rate distortion to study filters in a fixed universe U, with i.i.d. membership (see section 3.1), and a predetermined ratio between the two types of error ε_P and ε_N. We incorporated this view and modified all three aspects, in order to obtain a lower bound explicitly in ε_P and ε_N, which applies to filters on a fixed set.

Incremental filters. This is a more restrictive setting where the (one-sided) filters must support insertion from empty to at most n keys, where n is known before-hand. Specifically, Lovett and Porat [22] showed that such a filter must use $C(\varepsilon)n\log\frac{1}{\varepsilon}$ bits to achieve an FPR of ε for some $C(\varepsilon) > 1$. A corollary is that incremental filters (and dynamic filters which supports deletion) can never hope to be strongly optimal in the generic membership filter sense. The lower bound in this setting has yet to be exactly characterized, and there is a dynamic filter that uses $(1+o(1))(n\log\frac{1}{\varepsilon} + n\log_2 e)$ bits for fixed ε [4].

2.4.4 Other data structures with two-sided errors

As mentioned, there are several other works attempting to trade-off FPR with FNR on the space-suboptimal Bloom filter structure. We can view them as heuristics.

Retouched Bloom filter. [12] In this data structure, bits of the Bloom filter are reset to zero either randomly (to decrease FPR) or selectively (to avoid selected false positives). From the view of trading-off FPR and FNR, the former is rather trivial – in fact, it performs worse than selective insertion.

Generalized Bloom filter. [21] Here, the authors used a Bloom filter with some hash functions setting bits to 1, and other functions setting bits to 0, on every insertion of a key. Queries are tested accordingly using the same functions. This yields an upper bound on FPR that's independent of the number of keys inserted.

Compacted Bloom filter. [25] This design divides the standard Bloom filter's bit vector into blocks, and compressed each blocks into fewer bits. Insertions and queries were modified accordingly, with the goal of reducing space usage and FPR, at the expense of nonzero FNR.

Autoscaling Bloom filter. [20] This recent work uses a counting Bloom filter [16] to record how many times each bit in a Bloom filter is hashed into, and then discard the bits that have load smaller than some threshold T. The goal is to trade-off FPR with FNR to maximize *overall accuracy*, which is very similar to our topic of study.

3 Space Lower Bound for Two-Sided Filters

In this section, we prove the main lower-bound result Theorem 1. As before, all logarithms are base 2. Let $D_{KL}(p\|q)$ be the binary KL-divergence of a Bern(p) distribution from a Bern(q) distribution:

$$D_{KL}(p\|q) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}.$$

[3]In Bloom filter's case, there could even be an extreme case where all bits are set to 1 and $\varepsilon = 1$, albeit unlikely.

We first give a high-level overview of the proof, which follows from the oberservation that a filter is a *lossy code* that encodes the membership information (whether $x \in S$ or not) for each $x \in U$. Prior work [19] considered the most natural setting for applying information theoretical tools, which is somewhat different from the filter setting:

1. Every $x \in U$ has a probability p of being in S a priori.
2. Alice and Bob agree on a filter construction. Alice will use the filter to store S and send the filter to Bob, who will query every elemnt to decode all the membership information.
3. The error, or distortion, is measured by the percentage of membership information that Bob incorrectly decodes.

To connect this problem to the filter setting we need three steps. In section 3.1, we give an overview of the lossy compression for the i.i.d. setting, while proving a generalized version of the rate-distortion theorem for two error metrics. Then in section 3.2, we reduce the setting where S is a set of fixed size n to the case where S has i.i.d. membership with probability $p = \frac{n}{u}$, in the sence that both settings share similar lower bounds. Finally, in section 3.3, we study the rate-distortion function for sufficiently small p and obtain the desired lower bound.

3.1 Lossy compression of i.i.d. membership information

Consider the i.i.d. membership setting for the moment. Namely, now each $x \in U$ has a $p = \frac{n}{u}$ probability of being contained in S, independent from all other elements. Fixing p and taking $u \to \infty$, we can formulate the construction of filter into the following *lossy compression* problem: (see e.g. chapter 10 of [10])

1. Alice observes i.i.d. random variables $X_1, \ldots, X_u \sim \mathrm{Bern}(p)$. In our context, $X_i = \mathbb{1}\{x_i \in S\}$.
2. Alice compresses the outcome $\{X_i\}_{i=1}^u$ using an R-bit (random) encoding function $f_u : \{0,1\}^u \to \{0,1\}^{uR}$. R is called the **rate**, or the number of bits transmitted per element.
3. Bob uses (random) decoding function $g_u : \{0,1\}^{uR} \to \{0,1\}^u$ to recover approximations of the membership information $\{\hat{X}_i\}_{i=1}^u$, by taking
$$(\hat{X}_1, \ldots, \hat{X}_u) = g_u(f_u(X_1, \ldots, X_u)).$$

The **distortion** is defined by a custom measure of distance between the $(X_i, \hat{X}_i)$ pairs. In contrary to the textbook setting, we will use two (instead of one) distortion functions to measure the false positive and false negative rates simultaneously:

$$d_P(x, \hat{x}) = \begin{cases} \frac{1}{1-p} & \text{if } (x, \hat{x}) = (0,1) \\ 0 & \text{otherwise.} \end{cases} \qquad d_N(x, \hat{x}) = \begin{cases} \frac{1}{p} & \text{if } (x, \hat{x}) = (1,0) \\ 0 & \text{otherwise.} \end{cases}$$

Given the distortion measures, the rate-distortion functions are defined as follows:

Definition 8. For any $u \in \mathbb{N}$ and $p \in (0,1)$, the **rate-distortion function** $R_{p,u}(\varepsilon_P, \varepsilon_N)$ indicates the minimum per-element rate for codes with expected distortion at most $(\varepsilon_P, \varepsilon_N)$:

$$R_{p,u}(\varepsilon_P, \varepsilon_N) := \min \Bigg\{ R : \exists R\text{-bit functions } f_u, g_u, \text{ such that } \mathbb{E}\left[\frac{1}{u}\sum_{i=1}^{u} d_P(X_i, \hat{X}_i)\right] \le F_P, \\ \mathbb{E}\left[\frac{1}{u}\sum_{i=1}^{u} d_N(X_i, \hat{X}_i)\right] \le F_N \Bigg\}.$$

Meanwhile, for each p, the **information rate-distortion function** $R_p^{(I)}(\varepsilon_P, \varepsilon_N)$ is:

$$R_p^{(I)}(\varepsilon_P, \varepsilon_N) := \min\{I(X; \hat{X}) : X \sim \text{Bern}(p), \mathbb{E}[d_P(X, \hat{X})] \leq \varepsilon_P, \mathbb{E}[d_N(X, \hat{X})] \leq \varepsilon_N\},$$

where the minimum is taken over all joint distributions of $(X, \hat{X})$.

We prove the following two lemmas to characterize the rate-distortion functions in our setting. Their proofs can be found in appendix A.1. The first explicitly characterizes $R_p^{(I)}(\varepsilon_P, \varepsilon_N)$, and the second generalizes a central result in rate distortion theory.

Lemma 9. *Let $p \in (0,1)$. If $\varepsilon_P + \varepsilon_N \geq 1$, then $R_p^{(I)}(\varepsilon_P, \varepsilon_N) = 0$. Otherwise, we have:*

$$R_p^{(I)}(\varepsilon_P, \varepsilon_N) = H(p(1-\varepsilon_N) + (1-p)\varepsilon_P) - pH(1-\varepsilon_N) - (1-p)H(\varepsilon_P),$$

where $H(p)$ is the binary entropy of a Bern(p) *random variable.*

Proof. Because $X \in \{0,1\}$, we can characterize the conditional distribution $\hat{X} \mid X$ with two parameters:

$$\begin{cases} \lambda & := \mathbb{P}[\hat{X} = 1 \mid X = 0], \\ \mu & := \mathbb{P}[\hat{X} = 1 \mid X = 1]. \end{cases}$$

Let $\hat{X}(\lambda, \mu)$ denote the random variable whose conditional distribution is as defined above. Then, by standard results in information theory,

$$\begin{aligned} I(X; \hat{X}(\lambda, \mu)) &= H(\hat{X}(\lambda, \mu)) - H(\hat{X}(\lambda, \mu) \mid X) \\ &= H(p\mu + (1-p)\lambda) - pH(\mu) - (1-p)H(\lambda). \end{aligned}$$

The last step is to take minimum mutual information over all (λ, μ) which satisfy the distortion constraints, namely:

$$\begin{cases} \mathbb{E}[d_P(X, \hat{X}(\lambda, \mu))] = \lambda & \leq \varepsilon_P, \text{ and} \\ \mathbb{E}[d_N(X, \hat{X}(\lambda, \mu))] = 1 - \mu & \leq \varepsilon_N. \end{cases}$$

Solving the optimization problem:

$$\begin{aligned} \min \quad & I(X; \hat{X}(\lambda, \mu)) = H(p\mu + (1-p)\lambda) - pH(\mu) - (1-p)H(\lambda), \\ \text{s.t.} \quad & \begin{cases} \lambda - \varepsilon_P & \leq 0, \\ 1 - \mu - \varepsilon_N & \leq 0, \end{cases} \end{aligned}$$

□

Lemma 10. *For all $p \in (0,1)$ and $u \in \mathbb{N}$, we have:*

$$R_{p,u}(\varepsilon_P, \varepsilon_N) \geq R_p^{(I)}(\varepsilon_P, \varepsilon_N).$$

Proof. See appendix A.1. □

3.2 From fixed key density to i.i.d. membership

When the element set S is generated in the i.i.d. fashion above, $|S|$ follows a Binom(u, p) distribution, which is closer to a Poisson(n) distribution instead of being a fixed number n. However, using the law of large numbers, we show that this difference is negligible as $n \to \infty$.

Lemma 11. *For all $\varepsilon_P, \varepsilon_N \in (0,1)$ such that $\varepsilon_P + \varepsilon_N < 1$, and for all rational $p \in (0,1)$, the function L in Theorem 1 must satisfy:*

$$\liminf_{n \to \infty} \frac{1}{n} L(n, \varepsilon_P, \varepsilon_N) \geq \frac{1}{p} R_p^{(I)}(\varepsilon_P, \varepsilon_N).$$

Proof. Fix some n_1, u_1 with $\frac{n_1}{u_1} = p$, and define sequences $\{n_j\}, \{u_j\}$ to be $n_j = jn_1$ and $u_j = ju_i$, for all $j \in \mathbb{N}$. By law of large numbers, there is sequence of real numbers $\{\delta_j\}$ such that $\delta_j \searrow 0$, and for each j we have

$$\mathbb{P}\big[|S| \in [(1-\delta_j)n_j, (1+\delta_j)n_j]\big] \geq 1 - \delta_j.$$

We can hence construct the following encoding function:

1. If $|S| \in [(1-\delta_j)n_j, (1+\delta_j)n_j]$, then we build a filter of the entire S, using at most $L((1+\delta_j)n_j, \varepsilon_P, \varepsilon_N)$ bits.
2. If $|S| > (1+\delta_j)n_j$, then we send a dummy message indicating $\hat{X}_i = 1$ for all i.
3. If $|S| < (1-\delta_j)n_j$, then we send a dummy message indicating $\hat{X}_i = 0$ for all i.

Note that cases 2 and 3 incurs false negative distortion $\leq \frac{1}{p}$ and false positive distortion $\leq \frac{1}{1-p}$ respectively. Thus, we can bound the expected distortions of this protocol by:

$$\begin{cases} \mathbb{E}\left[\frac{1}{u_j}\sum_{i=1}^{u_j} d_P(X_i, \hat{X}_i)\right] \leq (1-\delta_j)\cdot\frac{1-(1-\delta_j)p}{1-p}\varepsilon_P + \delta_j \cdot \frac{1}{1-p} & =: a_j, \\ \mathbb{E}\left[\frac{1}{u_j}\sum_{i=1}^{u_j} d_N(X_i, \hat{X}_i)\right] \leq (1-\delta_j)\cdot\frac{(1+\delta_j)p}{p}\varepsilon_N + \delta_j \cdot \frac{1}{p} & =: b_j, \end{cases}$$

where $a_j \to \varepsilon_P$ and $b_j \to \varepsilon_N$ as $j \to \infty$. By lemma 9 and lemma 10, we have:

$$\frac{1}{u_j}L((1+\delta_j)n_j, \varepsilon_P, \varepsilon_N) \geq R_p^{(I)}(a_j, b_j) \text{ for all } j \in \mathbb{N}.$$

Now let $n \in \mathbb{N}$, then for some j we can squeeze it by $(1+\delta_j)n_j \leq n \leq (1+\delta_j)n_{j+1}$. It follows that:

$$\begin{aligned} \frac{L(n, \varepsilon_P, \varepsilon_N)}{n} &\geq \frac{L((1+\delta_j)n_j, \varepsilon_P, \varepsilon_N)}{(1+\delta_j)n_{j+1}} \\ &\geq \frac{u_j}{(1+\delta_j)n_{j+1}} R_p^{(I)}(a_j, b_j). \end{aligned}$$

As $n \to \infty$, we have $\frac{u_j}{(1+\delta_j)n_{j+1}} \to \frac{1}{p}$, $a_j \to \varepsilon_P$, and $b_j \to \varepsilon_N$. The desired claim then follows from the continuity of $R_p^{(I)}$. $\square$

3.3 Putting it all together

Now we are ready to prove the space lower bound for two-sided filters:

Proof for theorem 1. By lemma 11, we have:

$$\liminf_{n\to\infty} \frac{L(n, \varepsilon_P, \varepsilon_N)}{n} \geq \sup_{p \in (0,1)\cap\mathbb{Q}} \frac{1}{p} R_p^{(I)}(\varepsilon_P, \varepsilon_N),$$

where we can WLOG replace $p \in (0,1) \cap \mathbb{Q}$ by $p \in (0,1)$ in the supremum, since $\mathbb{Q}$ is dense in $(0,1)$ and $R_p^{(I)}$ is continuous in p.

Plugging in the expression for $R_p^{(I)}$, the right hand side equals:

$$\begin{aligned} \sup_p \frac{1}{p} R_p^{(I)}(\varepsilon_P, \varepsilon_N) &= \sup_p \frac{1}{p}\Big[H\big(p(1-\varepsilon_N) + (1-p)\varepsilon_P\big) - pH(1-\varepsilon_N) - (1-p)H(\varepsilon_P)\Big] \\ &= H(\varepsilon_P) - H(1-\varepsilon_N) + \underbrace{\sup_p \frac{1}{p}\Big[H\big(\varepsilon_P + p(1-\varepsilon_N-\varepsilon_P)\big) - H(\varepsilon_P)\Big]}_{\text{maximized when } p\to 0\text{, since } H(p) \text{ is concave}} \\ &= H(\varepsilon_P) - H(1-\varepsilon_N) + \lim_{p\to 0}\frac{1}{p}\Big[H\big(\varepsilon_P + p(1-\varepsilon_N-\varepsilon_P)\big) - H(\varepsilon_P)\Big] \\ &\qquad \text{(first-order in } p\text{: letting } c := 1-\varepsilon_N-\varepsilon_P,\ H(\varepsilon_P + pc) = H(\varepsilon_P) + pc\,H'(\varepsilon_P) + O(p^2)) \\ &= H(\varepsilon_P) - H(1-\varepsilon_N) + (1-\varepsilon_N-\varepsilon_P)H'(\varepsilon_P). \end{aligned}$$

Plugging in $H(p) = -p\log p - (1-p)\log(1-p)$ and $H'(p) = \log\frac{1-p}{p}$, this expands into:

$$\begin{aligned}\sup_p \frac{1}{p} R_p^{(I)}(\varepsilon_P, \varepsilon_N) &= -\varepsilon_P \log \varepsilon_P - (1-\varepsilon_P)\log(1-\varepsilon_P) + \varepsilon_N \log \varepsilon_N \\ &\quad + (1-\varepsilon_N)\log(1-\varepsilon_N) + (1-\varepsilon_P-\varepsilon_N)\log\frac{1-\varepsilon_P}{\varepsilon_P} \\ &= \varepsilon_N \log \frac{\varepsilon_N}{1-\varepsilon_P} + (1-\varepsilon_N)\log\frac{1-\varepsilon_N}{\varepsilon_P} \\ &= D_{KL}(1-\varepsilon_N \| \varepsilon_P).\end{aligned}$$

□

4 A Strongly Optimal Filter Construction

In this section, we present a filter design that can achieve the space lower bound of $D_{KL}(1 - \varepsilon_N\|\varepsilon_P) + o(1)$ per element. Our construction is inspired by Porat [29]. Although impractical, this filter is conceptually simple and serves to complement the lower bound result.

Theorem 12. *There exists a two-sided filter algorithm $\mathcal{A}$ that, given access to an infinitely long tape of random bits, is strongly optimal when $\varepsilon_P = 2^{-k}$ for some $k \in \mathbb{N}$.*

Proof. Let $\mathbb{F}_{2^k}$ denote the finite field with 2^k elements. Given U, we use the random tape to implement random oracles $f, h_1, h_2, \ldots : U \to (\mathbb{F}_{2^k})^n$, where $n = |S|$ is the number of keys. We call the function f *fingerprint function*, and the oracles $h_1, h_2, \ldots$ *hash functions.* We note that for each $x \in U$, the value $h_i(x)$ is a n-dimensional vector, where the scalar field of the vector space $(\mathbb{F}_{2^k})^n$ is just $\mathbb{F}_{2^k}$. On input $S = \{x_1, \ldots, x_n\}, \varepsilon_N, \varepsilon_P$, the algorithm does the following:

1.

Generate random vector $b \in (\mathbb{F}_{2^k})^n$ from the random tape, and we define $\mathcal{A}$ as follows:

1. Compute the values $N_j = \sum_{i=1}^n \mathbb{1}\{h_j(x_i) \cdot b = f(x_i)\}$ for $j \in [m]$, where m is specified later.

2. Pick the j with the largest N_j, namely $j^* = \arg\max_{j\in[m]} N_j$.

3. The filter handles query $x \in U$ by outputting the indicator $\mathbb{1}\{f(x) = h_{j^*}(x)\}$.

4. We can store the value j^* with $\log m$ bits. Instructions for implementing $f, \{h_j\}$, and b from the random tape can also be stored using $o(n)$ bits.

We now choose the parameter m. Note that $\mathbb{1}\{h_j(x_i) \cdot b = f(x_i)\}$ is a Bernoulli random variable with success probability 2^{-k}, independent of any other such indicator. Thus we have $N_j \sim \text{Binom}(n, 2^{-k})$, which, for $1 - \varepsilon_N > 2^{-k}$, satisfies the following:

$$\mathbb{P}[N_j \geq (1-\varepsilon_N)n] = 2^{-n(D_{KL}(1-\varepsilon_N\|2^{-k})+o(1))}.$$

We now define $m := \min_{m'\in\mathbb{N}}\{m' : \mathbb{E}\left[\max_{j\in[m']} N_j\right] \geq (1-\varepsilon_N)n\}$. Then, by definition the resulting filter has expected FNR of at most ε_N. The proof would be finished if $\log m = D_{KL}(1-\varepsilon_N\|2^{-k}) + o(1)$, which we now show.

Suppose otherwise, that $\log m \geq n(D_{KL}(1-\varepsilon_N\|2^{-k}) + \Omega(1))$. Then, by the binomial tail bound there must be some m_0, which also depends on n, such that:

$$\mathbb{P}[N_j \geq (1-\varepsilon_N+\delta)n] < m_0 < m, \text{ for some } \delta > 0, \text{ for all large enough } n.$$

It follows that:

$$\begin{aligned}\mathbb{E}[\max_{j\in[m_0]} N_j] &\geq \mathbb{P}[\max_{j\in[m_0]} N_j \geq (1-\varepsilon_N+\delta)n]\cdot(1-\varepsilon_N+\delta)n+0\\ &= (1-o(1))\cdot(1-\varepsilon_N+\delta)n,\end{aligned}$$

which is at least $(1-\varepsilon_N)n$ for sufficiently large n. This contradicts the assumption that m is the smallest number such that $\mathbb{E}[\max_{j\in[m]} N_j] \geq (1+\varepsilon_N)n$. $\square$

5 Aggregating Multiple Filters

Our final result is on the performance of *aggregating* the answers of multiple filers. Consider constructing a filter $\mathcal{F}$ for S with error rates $(\varepsilon_P, \varepsilon_N)$, using the following procedure:

1. Based on M, calculate parameters $\varepsilon_{P,M}, \varepsilon_{N,M}$ and $T\in\mathbb{N}$ with methods specified later.
2. On initialization, initialize M sub-filters $\mathcal{F}_1,\ldots,\mathcal{F}_M$ for S with error rates $(\varepsilon_{P,M}, \varepsilon_{N,M})$, independently.
3. When queried $x\in U$, we perform the same query on every $\mathcal{F}_i$ for $i\in[M]$.
4. Return 1 if at least T out of the M filters return 1.

An analogous structure exists for one-sided filters: since no false negatives are present, one can simply output 1 iff every sub-filter $\mathcal{F}_i$ outputs 1. The resulting filter $\mathcal{F}$ has FPR of $\varepsilon_{P,M}^M = \varepsilon_P$ and space usage $Mn\log\frac{1}{\varepsilon_{P,M}} = n\log\frac{1}{\varepsilon_P}$. We thus conclude that the one-sided filter constructed by aggregating strongly optimal one-sided filters is still strongly optimal, regardless of M.

We show that the same is *not true* for general filers. But first, we specify the parameters $(\varepsilon_{P,M}, \varepsilon_{N,M}, T)$ based on $(\varepsilon_P, \varepsilon_N, M)$.

References

[1] Djamal Belazzougui, Fabiano C. Botelho, and Martin Dietzfelbinger. Hash, displace, and compress. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 682–693. Springer, 2009.

[2] Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, and Shikha Singh. Bloom filters, adaptivity, and the dictionary problem. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 182–193. IEEE Computer Society, 2018.

[3] Michael A. Bender, Martin Farach-Colton, Rob Johnson, Russell Kraner, Bradley C. Kuszmaul, Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P. Spillane, and Erez Zadok. Don't thrash: How to cache your hash on flash. *Proc. VLDB Endow.*, 5(11):1627–1637, 2012.

[4] Michael A. Bender, Martin Farach-Colton, John Kuszmaul, William Kuszmaul, and Mingmou Liu. On the optimal time/space tradeoff for hash tables. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1284–1297. ACM, 2022.

[5] Ioana O. Bercea and Guy Even. A dynamic space-efficient filter with constant time operations. In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020, June 22-24, 2020, Tórshavn, Faroe Islands*, volume 162 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[6] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[7] Alexander Dodd Breslow and Nuwan Jayasena. Morton filters: Faster, space-efficient cuckoo filters via biasing, compression, and decoupled logical sparsity. *Proc. VLDB Endow.*, 11(9):1041–1055, 2018.

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

[9] Larry Carter, Robert W. Floyd, John Gill, George Markowsky, and Mark N. Wegman. Exact and approximate membership testers. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 59–65. ACM, 1978.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley, 2001.

[11] Martin Dietzfelbinger and Rasmus Pagh. Succinct data structures for retrieval and approximate membership (extended abstract). In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 2008.

[12] Benoit Donnet, Bruno Baynat, and Timur Friedman. Retouched bloom filters: allowing networked applications to trade off selected false positives against false negatives. In Christophe Diot and Mostafa H. Ammar, editors, *Proceedings of the 2006 ACM Conference on Emerging Network Experiment and Technology, CoNEXT 2006, Lisboa, Portugal, December 4-7, 2006*, page 13. ACM, 2006.

[13] David Eppstein. Cuckoo filter: Simplification and analysis. In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 8:1–8:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[14] Tomer Even, Guy Even, and Adam Morrison. Prefix filter: Practically and theoretically better than bloom. *Proc. VLDB Endow.*, 15(7):1311–1323, 2022.

[15] Bin Fan, David G. Andersen, Michael Kaminsky, and Michael Mitzenmacher. Cuckoo filter: Practically better than bloom. In Aruna Seneviratne, Christophe Diot, Jim Kurose, Augustin Chaintreau, and Luigi Rizzo, editors, *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, CoNEXT 2014, Sydney, Australia, December 2-5, 2014*, pages 75–88. ACM, 2014.

[16] Li Fan, Pei Cao, Jussara M. Almeida, and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In Gerald Neufeld, Gary S. Delp, Jonathan Smith, and Martha Steenstrup, editors, *Proceedings of the ACM SIGCOMM 1998 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 31 - September 4, 1998, Vancouver, B.C., Canada*, pages 254–265. ACM, 1998.

[17] Thomas Mueller Graf and Daniel Lemire. Xor filters. *ACM J. Exp. Algorithmics*, 25:1–16, 2020.

[18] Thomas Mueller Graf and Daniel Lemire. Binary fuse filters: Fast and smaller than xor filters. *ACM J. Exp. Algorithmics*, 27:1.5:1–1.5:15, 2022.

[19] Paul Hurley and Marcel Waldvogel. Bloom filters: One size fits all? In *32nd Annual IEEE Conference on Local Computer Networks (LCN 2007), 15-18 October 2007, Clontarf Castle, Dublin, Ireland, Proceedings*, pages 183–190. IEEE Computer Society, 2007.

[20] Denis Kleyko, Abbas Rahimi, Ross W. Gayler, and Evgeny Osipov. Autoscaling bloom filter: controlling trade-off between true and false positives. *Neural Comput. Appl.*, 32(8):3675–3684, 2020.

[21] Rafael P. Laufer, Pedro B. Velloso, and Otto Carlos Muniz Bandeira Duarte. A generalized bloom filter to secure distributed network applications. *Comput. Networks*, 55(8):1804–1819, 2011.

[22] Shachar Lovett and Ely Porat. A lower bound for dynamic approximate membership data structures. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 797–804. IEEE Computer Society, 2010.

[23] Lailong Luo, Deke Guo, Richard T. B. Ma, Ori Rottenstreich, and Xueshan Luo. Optimizing bloom filter: Challenges, solutions, and comparisons. *IEEE Commun. Surv. Tutorials*, 21(2):1912–1949, 2019.

[24] Páll Melsted and Jonathan K. Pritchard. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinform.*, 12:333, 2011.

[25] Negar Mosharraf, Anura P. Jayasumana, and Indrakshi Ray. Compacted bloom filter. In *2nd IEEE International Conference on Collaboration and Internet Computing, CIC 2016, Pittsburgh, PA, USA, November 1-3, 2016*, pages 304–311. IEEE Computer Society, 2016.

[26] Anna Pagh, Rasmus Pagh, and S. Srinivasa Rao. An optimal bloom filter replacement. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 823–829. SIAM, 2005.

[27] Rasmus Pagh and Flemming Friche Rodler. Lossy dictionaries. In Friedhelm Meyer auf der Heide, editor, *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, volume 2161 of *Lecture Notes in Computer Science*, pages 300–311. Springer, 2001.

[28] Prashant Pandey, Alex Conway, Joe Durie, Michael A. Bender, Martin Farach-Colton, and Rob Johnson. Vector quotient filters: Overcoming the time/space trade-off in filter design. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 1386–1399. ACM, 2021.

[29] Ely Porat. An optimal bloom filter replacement based on matrix solving. In Anna E. Frid, Andrey Morozov, Andrey Rybalchenko, and Klaus W. Wagner, editors, *Computer Science - Theory and Applications, Fourth International Computer Science Symposium in Russia, CSR 2009, Novosibirsk, Russia, August 18-23, 2009. Proceedings*, volume 5675 of *Lecture Notes in Computer Science*, pages 263–273. Springer, 2009.

[30] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[31] Ori Rottenstreich and Isaac Keslassy. The bloom paradox: When not to use a bloom filter. *IEEE/ACM Trans. Netw.*, 23(3):703–716, 2015.

[32] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Commun. Surv. Tutorials*, 14(1):131–155, 2012.

[33] Minmei Wang, Mingxun Zhou, Shouqian Shi, and Chen Qian. Vacuum filters: More space-efficient and faster replacement for bloom and cuckoo filters. *Proc. VLDB Endow.*, 13(2):197–210, 2019.

A Appendix

A.1 Proof for Lemma10

Lemma 13 (Same as Lemma 10)**.** *For all $p \in (0,1)$ and $u \in \mathbb{N}$, we have:*

$$R_p^u(\varepsilon_P, \varepsilon_N) \geq R_p^{(I)}(\varepsilon_P, \varepsilon_N).$$

Before the proof itself, we first show the useful fact that $R_p^u(\varepsilon_P, \varepsilon_N)$ is jointly convex in both inputs.

Lemma 14. *For all $u \in \mathbb{N}, p \in (0,1)$, inputs $\varepsilon_P, \varepsilon'_P, \varepsilon_N, \varepsilon'_N \in [0,1]$, and $\lambda \in (0,1)$, we have:*

$$R_{p,u}(\lambda\varepsilon_P + (1-\lambda)\varepsilon'_P, \lambda\varepsilon_N + (1-\lambda)\varepsilon'_N) \leq \lambda R_{p,u}(\varepsilon_P, \varepsilon_N) + (1-\lambda)R_{p,u}(\varepsilon'_P, \varepsilon'_N)$$

Proof. Fixing X, let $\hat{X}$ and $\hat{X}'$ be random variables achieving the optimal rates:

$$\begin{cases} I(X;\hat{X}) &= R_{p,u}(\varepsilon_P, \varepsilon_N), \\ I(X;\hat{X}') &= R_{p,u}(\varepsilon'_P, \varepsilon'_N). \end{cases}$$

Now consider random variable $\hat{X}_\lambda$ defined by:

$$\hat{X}_\lambda = \begin{cases} \hat{X} \text{ with probability } \lambda, \\ \hat{X}' \text{ with probability } 1-\lambda, \end{cases}$$

independently from $X, \hat{X}, \hat{X}'$. From linearity of expectation, we have:

$$\begin{cases} \mathbb{E}[d_P(X,\hat{X}_\lambda)] &\leq \lambda\varepsilon_P + (1-\lambda)\varepsilon'_P, \\ \mathbb{E}[d_N(X,\hat{X}_\lambda)] &\leq \lambda\varepsilon_N + (1-\lambda)\varepsilon'_N. \end{cases}$$

Since $I(X;\hat{X})$ is convex in the conditional distribution $\hat{X} \mid X$, the desired statement follows from:

$$\begin{aligned} R_{p,u}(\lambda\varepsilon_P + (1-\lambda)\varepsilon'_P, \lambda\varepsilon_N + (1-\lambda)\varepsilon'_N) &\leq I(X;\hat{X}_\lambda) \\ &\leq \lambda I(X;\hat{X}) + (1-\lambda)I(X;\hat{X}') \\ &= \lambda R_{p,u}(\varepsilon_P, \varepsilon_N) + (1-\lambda)R_{p,u}(\varepsilon'_P, \varepsilon'_N). \end{aligned}$$

□

Now we are ready for the proof. We use the following shorthand for the tuples of random variables: $X^u = (X_1, \ldots, X_u)$ and $\hat{X}^u = (\hat{X}_1, \ldots, \hat{X}_u)$.

Proof for Lemma 10. For all R such that there exists pair of (random) functions $f : \{0,1\}^u \to \{0,1\}^{uR}$ and $g : \{0,1\}^{uR} \to \{0,1\}^u$ satisfying distortion requirements $\varepsilon_P, \varepsilon_N$, we must have:

$$\begin{aligned}
uR &\geq H(f(X^u)) \\
&\geq I(X^u; f(X^u)) \\
&\geq I(X^u; \hat{X}^u) \\
&= H(X^u) - H(X^u \mid \hat{X}^u) \\
&= \sum_{i=1}^{u} H(X_i) - \sum_{i=1}^{u} H(X_i \mid \hat{X}^u, X_1, \ldots, X_{i-1}) \\
&\geq \sum_{i=1}^{u} H(X_i) - \sum_{i=1}^{u} H(X_i \mid \hat{X}_i) \\
&= \sum_{i=1}^{u} I(X_i; \hat{X}_i) \\
&\geq \sum_{i=1}^{u} R_{p,u}(\mathbb{E}[d_P(X_i, \hat{X}_i)], \mathbb{E}[d_N(X_i, \hat{X}_i)]) \\
&\geq uR_p^u \left(\mathbb{E}\left[\frac{1}{u} \sum_{i}^{u} d_P(X_i, \widehat{X}_i) \right], \mathbb{E}\left[\frac{1}{u} \sum_{i}^{u} d_N(X_i, \hat{X}_i) \right] \right) \text{ by convexity,} \\
&\leq uR_{p,u}\left(\varepsilon_P, \varepsilon_N\right).
\end{aligned}$$

□

A.2 Aggregating filters

Recall that we want to build a aggregated filter $\mathcal{F}$ using M smaller and more erroneous sub-filters $\mathcal{F}_1, \ldots, \mathcal{F}_M$. On when queried $x \in U$, $\mathcal{F}$ would make the same query x to the sub-filters uses the M, and it outputs 1 iff at least T of the M sub-filters output 1, where T is a threshold to be determined. We are interested in the necessary space for $\mathcal{F}$ to achieve error level $(\varepsilon_P, \varepsilon_N)$, as $M \to \infty$. Namely, is it space-efficient to use a large number of erroneous sub-filters, comparing to simply building one big filter?

From a statistical test point of view, by Neyman-Pearson lemma, the family of thresholding methods (for different choices of T) should the optimal methods of aggregating the outputs of independent filter.

Let threshold T be proportional to the number of sub-filters, parameterized by $t = \frac{T}{M} \in (\varepsilon_P, 1 - \varepsilon_N)$, while we take the limit $M \to \infty$. For each M, we can first determine the necessary error rates $(\varepsilon_P, \varepsilon_N)$ of the sub-filters, in order for the aggregated filter $\mathcal{F}$ to have error rates $(\varepsilon_P, \varepsilon_N)$. Intuitively, as $M \to \infty$, the sub-filters are allowed to be more erroneous, and we have $\varepsilon_{P,M}$ approaches t from below, and $1 - \varepsilon_{N,M}$ approaches t from above. We now quantify this argument.

For convenience, we assume the sub-filters satisfy at most $(\varepsilon_P, \varepsilon_N)$ error rate on *every* element, over the randomness of random tape r. This is a stronger assumption than strong optimality, but our conclusion is unaffected. Specifically, we assume:

$$\begin{cases} \mathbb{P}_r[\mathcal{F}_i(x) = 1] = \varepsilon_{P,M}, & \text{for each non-key } x \in U \setminus S, \\ \mathbb{P}_r[\mathcal{F}_i(x) = 0] = \varepsilon_{N,M}, & \text{for each key } x \in S. \end{cases}$$

In this setting, by the independence of the sub-filters $\mathcal{F}_i$, we have $\mathcal{F}(x) \sim \text{Binom}(M, \varepsilon_{N,M})$ for keys $x \in S$, and $\mathcal{F}(x) \sim \text{Binom}(M, \varepsilon_{P,M})$ for non-keys $x \in U \setminus S$. By Binomial tail bounds, we have:

$$\begin{cases} \frac{1}{\sqrt{2M}} 2^{-M\cdot D_{KL}(t\|\varepsilon_{P,M})} \le \varepsilon_P \le 2^{-M\cdot D_{KL}(t\|\varepsilon_{P,M})}, & \text{for each non-key } x \in U \setminus S, \\ \frac{1}{\sqrt{2M}} 2^{-M\cdot D_{KL}(t\|1-\varepsilon_{N,M})} \le \varepsilon_N \le 2^{-M\cdot D_{KL}(t\|1-\varepsilon_{N,M})}, & \text{for each key } x \in S. \end{cases}$$

Given t, it is thus necessary and sufficient to set $\varepsilon_{P,M}$ and $\varepsilon_{N,M}$ such that:

$$\begin{cases} D_{KL}(t\|\varepsilon_{P,M}) & = \frac{1}{M}\log\frac{1}{\varepsilon_P} - o(1) \\ D_{KL}(t\|1-\varepsilon_{N,M}) & = \frac{1}{M}\log\frac{1}{\varepsilon_N} - o(1). \end{cases}$$

Since $t \in (\varepsilon_{P,M}, 1-\varepsilon_{N,M})$, the total space usage of the M sub-filters is:

$$\begin{aligned} M \cdot D_{KL}(1-\varepsilon_{N,M}\|\varepsilon_{P,M}) &> M \cdot D_{KL}(t\|\varepsilon_{P,M}) \\ &= \log\frac{1}{\varepsilon_P} + o(M) \end{aligned}$$