Mitigating False Positives in Filters: to Adapt or to Cache?

Michael A. Bender[1] Rathish Das[1] Martín Farach-Colton[2] Tianchi Mo[1] David Tench[1]
Yung Ping Wang[2]

Abstract

A filter is adaptive if it achieves a false positive rate of ε on each query independently of the answers to previous queries. Many popular filters such as Bloom filters are not adaptive—an adversary could repeat a false-positive query many times to drive the false-positive rate to 1. Bender et al. [4] formalized the definition of adaptivity and gave a provably adaptive filter, the broom filter. Mitzenmacher et al. [20] gave a filter that achieves a lower empirical false-positive rate by exploiting repetitions.

We prove that an adaptive filter has a lower false-positive rate when the adversary is stochastic. Specifically, we analyze the broom filter against queries drawn from a Zipfian distribution. We validate our analysis empirically by showing that the broom filter achieves a low false-positive rate on both network traces and synthetic datasets, even when compared to a regular filter augmented with a cache for storing frequently queried items.

1 Introduction

A ***filter*** is a dictionary data structure that maintains a set $\mathcal{S} \subseteq \mathcal{U}$ of items under approximate membership queries, insertions, and sometimes deletions. If $|\mathcal{S}| = n$ and $|\mathcal{U}| = u$, then an error-free dictionary, which always answers membership queries correctly, requires $\Omega(n \log u)$ bits. In order to save space, a filter is allowed a one-sided false-positive error probability of ε: for any $x \in \mathcal{S}$, a filter must answer PRESENT, whereas for any $x \in \overline{\mathcal{S}} = \mathcal{U} \setminus \mathcal{S}$, a filter is allowed to answer PRESENT, but with probability at most ε and otherwise answers ABSENT. A filter requires $\Omega(n \log(1/\varepsilon))$ bits and there are filters, such as the Bloom filter [6, 7], and the quotient filter [5, 22], that match this space bound,[1] and even filters that match it up to low-order terms [3, 4, 21].

Bender et al. [4] point out that the false-positive rate of ε only applies to single queries. A sequence of queries can have a much higher false-positive rate if an adversary can repeat queries, because, for example, the adversary can make random queries on items from $\overline{\mathcal{S}}$ until it finds a false positive, and then it can repeat the false positive to obtain a false-positive rate of 1. In this case, the adversary does not have access to the random bits of the filter but knows the filter's output. But even this much information may not be needed, because in many settings, a filter's output is verified by querying a (slow) dictionary. Therefore, a timing attack may be enough for the adversary to drive the false positive rate to 1.

They define an ***adaptive filter*** to be a filter that guarantees a false-positive probability of at most ε for every query *regardless of the answers to previous queries*. They present the broom filter, which is an adaptive modification of the quotient filter. Defining adaptivity requires specifying what information is made available to the filter so that it can adapt. After each query response, the filter learns if that query resulted in a false positive, and the filter can change its internal representation based on this knowledge. Interestingly, after the filter has answered the query, the system can let the filter know which elements were false positives without increasing the asymptotic cost; see Section 2.[2]

Thus, for any given ε, there is a broom filter that maintains optimal space (up to lower-order terms) and constant time per operation. A broom filter adapts by extending fingerprints of items stored in the filter until a false positive is corrected. A side effect of extending fingerprints is that other false positives can be serendipitously corrected. An adversary can minimize the benefits of adaptivity by never repeating a query, but, as we will see, the effective false positive rate is below ε, even in this case.

[1]Stony Brook University, Stony Brook, NY 11794-2424, USA. Email: {bender, radas, timo, dtench}@cs.stonybrook.edu.

[2]Rutgers University, New Brunswick, NJ 08854, USA. Email: martin@farach-colton.com, lacy108m@gmail.com.

[1]The cuckoo filter [20] matches this bound empirically but not asymptotically, in that, as n grows, the probability that no hash function will successfully build a cuckoo filter of size $O(n \log(1/\varepsilon))$ goes to 1. However, before this failure condition—that is, pre-asymptotically—the cuckoo filter also uses $O(n \log(1/\varepsilon))$ bits.

[2]In fact, the system detects that an element x is a false positive by finding an element $y \in \mathcal{S}$ with which it collides in the filter. Providing the filter with y in order to aid its adaptivity also does not increase the asymptotic cost.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

It's not clear how a worst-case analysis can illustrate the benefits that repetition yields, since one worst-case analysis with repetitions is to have no repetitions at all. Thus, Bender et al. left open the question of how to quantify the advantage achievable by an adaptive filter in the presence of repetitions.

Mitzenmacher et al. [20] were the first to propose adapting a filter based on previous false positives. They observed that network traces often have repeated IP addresses and proposed a variant of the cuckoo filter that adapts heuristically. They empirically demonstrate that the adaptive cuckoo filter improves the observed false positive rate on network traces by between one and two orders of magnitude.

We note that Bruck et al. [8] also demonstrate the performance advantage that a filter can obtain when the input is drawn from a distribution. Specifically, they show that they can improve the false-positive probability of a filter by reducing the false-positive probability of queries that are known to be likely. Thus the construction of their weighted Bloom filter depends on the input distribution. Bloomier filters [10] can encode a set of elements from $\overline{S}$ on which the filter must answer ABSENT. However, neither weighted Bloom filters nor Bloomier filters adapt, both because they are static and because they require foreknowledge of the likely negative queries. Kopelowitz et al. [14] present an adaptive filter which optimizes running time by amortizing the cost of performing updates.

Results. In this paper, we quantify the improvement in the false-positive rate of an adaptive filter on stochastic sequences that are likely to include repetitions. Specifically, we model the query sequence by a Zipfian distribution. We show that, although repetitions can increase the false-positive rate of static filters [4], repetitions can reduce the false positive rate of adaptive filter.

In contrast to weighted bloom filters and Bloomier filters, we only use the input distribution to model query sequences with repetitions. Our reliance on Zipfians is analytical: the broom filter is oblivious to the source of its queries and retains its strong worst-case guarantees. So, for example, our results hold if the distribution changes over time, or if we have a mixture of distributional and adversarial inputs.

We prove bounds on the false-positive rate of a broom filter against an adversary that draws its negative queries from a Zipfian distribution on $\overline{S}$. The main characteristic of a Zipfian is its constant, s, where the ith most frequent item is drawn with probability proportional to $1/i^s$. Not surprisingly, when s is small (so the distribution is flat, with few repetitions), the false-positive rate of an adaptive filter approaches ε, and when s grows large, it approaches 0.

So it seems that adaptive filters deliver performance benefits when compared with a non-adaptive filter on repetitive sequences—but this conclusion is premature. Certainly our result implies that the false-positive rate of an adaptive filter is lower than the false-positive rate of a non-adaptive filter in highly repetitive query sequences. But this may not transfer to improved performance. Other parts of the system (such as OS caches) may reduce the cost of false positives. Suppose, for example that a dictionary is stored in slow storage and a filter is used to eliminate most negative queries to the dictionary. When the filter answers PRESENT, the system must verify the answer (and perhaps return the value associated with the key). If the pages needed to verify this answer are still in cache when this false positive-query is repeated, the verification can happen without accessing slow memory. So, even though a non-adaptive filter may not decrease its false-positive rate with repetitions, even in the presence of caches, the *cost* of a repeated false positive may be substantially reduced in this case, which might reduce the benefit we see of adaptive filters over non-adaptive ones.

We therefore compare the broom filter with a ***cache-augmented filter (CAF)***, which is a non-adaptive filter that caches false positives so that they are not repeated as long the false positive is not evicted from the cache. Since the broom filter uses n extra bits to adapt, we give the CAF n bits for its cache, so that it can store $n/\log u$ false positives.

We bound the theoretical performance of CAFs. Our results show that when the Zipfian constant is at most 1, which means that the query sequence has very few repetitions, both the broom filter and the CAF achieve a rate of approximately ε.[3] When s is much larger than 1, then both broom filters and CAFs do well because the sequence is so repetitive that a small cache suffices to handle almost all false positives. When s is modestly larger than 1, our theorems show that the performance of a filter is quite sensitive to the effective cache sizes: broom filters, which can cache n false positives, have a very low false positive rate, whereas CAFs, which have small caches, still achieve a substantial improvement over non-adaptive filters, but do not match the improvement in false positive rate offered by broom filters.

Our results on synthetic and network trace data strongly support our theoretical findings. In particular, the effective Zipfian constant for our network traces is in the $1.2 - 1.3$ range, and broom filters have effective false-positive rates that are 31-73% lower than that of CAFs and 90-97% lower than that of quotient filters. We also describe how a broom filter may serendipitously eliminate potential future false positives while adapting to an observed false positive. We then demonstrate empirically that this serendipitous correction effect accounts for 24-55% of all false positives avoided on network trace data.

We conclude that broom filters offer significant practical savings in terms of achievable false-positive rates on real-world data. The implementation of the broom filter is cur-

[3] We note that our analysis does not take into account the serendipitous false-positive corrections of a broom filter, so our bound of ε is an upper bound on the broom filter's false positive rate.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

rently quite slow, and in future work we hope to implement a broom filter that is as fast as a quotient filter and maintains its strong false-positive performance.

2 Filters, False Positives, and Zipf

Filters. In the previous literature, analytical results about the false-positive rates of filters have focused on oblivious adversaries versus static filters [3,5,6,10,21,22] or adaptive adversaries versus adaptive filters [4]. Here, we formalize the notion of an adaptive filter against an oblivious adversary and specify the different notions of false-positive rate. The full definition of the false-positive rate of an adaptive filter it more complicated than we need here, so our definition of adaptive filters is streamlined.

Definition 1 (Filter)**.** *A **filter instance on $\mathcal{S}$** is a deterministic function $f : \mathcal{U} \to \{0,1\}$ such that $x \in \mathcal{S} \Rightarrow f(x) = 1$. A **filter on set $\mathcal{S}$** is a pair $F = (\mathcal{F}, \Delta)$ where $\mathcal{F}$ is a set of filters instances on set $\mathcal{S}$ and Δ is a probability distribution on $\mathcal{F}$.*

Definition 2 (Adaptive Filter against an Oblivious Adversary)**.** *An **adaptive filter on set $\mathcal{S}$** is an online algorithm that, on query sequence $X = x_0, x_1, \ldots \in \overline{\mathcal{S}}^+$, after seeing $x_0, \ldots, x_{i-1}$, computes a filter $F_i = (\mathcal{F}_i, \Delta_i)$ on $\mathcal{S}$ that it uses to answer query x_i.*

This latter definition is a special case of the full definition of an adaptive filter [4]. First and importantly, this definition applies to an oblivious adversary, whereas the full definition of an adaptive filter generalizes to an adaptive adversary, where it takes the form of an interactive game between the filter and the adaptive adversary. Secondly, this definition only applies to filters where the set $\mathcal{S}$ is static. (This restriction is for convenience, and is not a substantial simplification.) Third, this filter only adapts based on true and false positives, and not on negatives. (Again, this restriction is for convenience and is easily generalized. But it is hardly a restriction at all since all known adaptive filters only adapt based on false positives.)

Both of these definitions are orthogonal to a cost model. To explain what we mean, recall how a filter is used in a system. In particular, the filter is a compact data structure that approximately summarizes a larger remote dictionary $\mathcal{D}$ representing $\mathcal{S}$, where by "remote" we mean "expensive to access". For a filter instance f and query x, each time that $f(x) = 0$, f "filters out" a remote access to $\mathcal{D}$; in contrast, if $f(x) = 1$, then $\mathcal{D}$ must be accessed to determine whether or not $x \in \mathcal{S}$. The main cost to be minimized is the number of (expensive) remote accesses to $\mathcal{D}$. After the remote access to $\mathcal{D}$, the filter can learn whether it had a false positive or a true positive, so that it can adapt. This remote access can also be used to transfer additional information at no extra (asymptotic) cost. For example, it costs one remote access to determine for each x for which $f(x) = 1$, whether or not $x \in \mathcal{S}$. If $x \notin \mathcal{S}$, then for the same cost, the remote dictionary can return a witness $y \in \mathcal{S}$ that collides with x in the filter.

Therefore, filters have a variety of costs. The filter must be small, but this limits the false-positive rate [9, 18]. But the false-positive rate must be as low as possible, because the primary objective is to minimize the number of remote accesses on $x \notin \mathcal{S}$, which happens on false positives. A secondary objective is to limit the (amortized or worst-case) CPU cost for all operations, such as answering queries and adapting. The broom filter is provably optimal in all these dimensions.

As we sketched in Section 1, the metric of remote accesses explains why we compare an adaptive filter to a cache-augmented filter (CAF). The CAF is motivated by the common practice of the system caching part of $\mathcal{D}$ in local memory, which adaptively filters out remote accesses to $\mathcal{D}$. Against an adaptive adversary, the CAF can be shown to perform poorly, but the question is how well it compares to an adaptive filter when the adversary is oblivious.

False-Positive Rate. In order to define the false-positive rate of a filter $F = (\mathcal{F}, \Delta)$, for all $x \in \mathcal{U}$ let

$$F(x) = \mathop{\mathbb{E}}_{f \sim \Delta}[f(x)].$$

For $X = x_0, \ldots, x_\ell \in \mathcal{U}^{\ell+1}$, let

$$F(X) = \sum_{i=0}^{\ell} F_i(x_i),$$

that is, the expected number of (true and false) positives that F generates on X. A static filter is a special case of an adaptive filter where $F_0 = F$, and where if f is the filter instance chosen to answer x_0, then $F_i = (\{f\}, \{\Delta_i(f) = 1\})$ for all $i \in [0, \ell]$. So $F(X)$ is defined for both static and adaptive filters.

Definition 3. *The **false-positive probability of filter F on $\mathcal{S}$** is*

$$\max_{x \in \overline{\mathcal{S}}} \{F(x)\}.$$

*On sequences, we define the **false-positive rate of filter F on $\mathcal{S}$** is*

$$\max_{X \in \overline{\mathcal{S}}^+} \left\{ \frac{F(X)}{|X|} \right\}.$$

Definition 4. *For any set $\mathcal{S}$, let $\mathcal{X}$ be a distribution on $\overline{\mathcal{S}}^+$. The **distributional false-positive rate of filter F on $\mathcal{X}$** is*

$$\mathop{\mathbb{E}}_{X \sim \mathcal{X}} \left[\frac{F(X)}{|X|} \right].$$

Let $\mathcal{Z}$ be a distribution on $\overline{\mathcal{S}}$, and let $\mathcal{Z}^\ell$ be the probability distribution of strings of length ℓ whose characters are

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

*drawn independently from $\mathcal{Z}$. The **distributional false-positive rate of filter F on $\mathcal{Z}$** is*

$$\inf_{\ell>0}\left\{\mathop{\mathbb{E}}_{X\sim\mathcal{Z}^\ell}\left[\frac{F(X)}{\ell}\right]\right\}.$$

For a static filter, the false-positive rate is the same for all lengths ℓ, and so, in particular, whatever the false-positive rate is on characters is the false-positive rate on strings. For adaptive filters, this condition need not hold, so we take the maximum false-positive rate over all string lengths.

Zipfian Sequences. We analyze the false-positive rate of filters on Zipfian sequences, so here we give a definition of Zipfian distributions and sequences.

Definition 5 (Empirical Zipfian)**.** *Let $\mathcal{M}$ be a multiset where the ith most frequent element in M has frequency c_i and the number of distinct items in $\mathcal{M}$ is m. Multiset $\mathcal{M}$ is **Zipfian of order s** if $c_i = 1/(i^s H_{s,m})$, for $i \in [1,m]$, and the normalization constant $H_{s,m} = \sum_{j=1}^{m} 1/j^s$.*

We will characterize sequences based on the multiset of characters comprising the sequence. In this paper, we consider network traces and compute the best-fit s to model their frequency distributions. We also analyze algorithms against Zipfian sources, so we model distributions on multisets as follows.

Definition 6 (Distributional Zipfian.)**.** *Let $\mathcal{A} = \{a_1,\ldots,a_m\}$ sorted in decreasing order by their probability in $\mathcal{Z}_{s,m}$, the Zipfian distribution of order s on $\mathcal{A}$. Let $M = s_1,\ldots,s_\ell$, where $s_i \sim \mathcal{Z}_{s,m}$, that is, a sequence of items drawn from a Zipfian. Then M is an **i.i.d. Zipfian sequence** and we say that $M \sim \mathcal{Z}^\ell_{s,m}$.*

Zipfian distributions are a standard way to model sequences that include repetitions [11–13,15–17,19,23,24,26], and here we bound the number of samples that need to be drawn in order to see x distinct items. For multiset $\mathcal{M}$, let $\|\mathcal{M}\|$ be the number of distinct items in $\mathcal{M}$. Let $D_{s,m}(\ell) = \mathbb{E}_{\mathcal{M}\sim\mathcal{Z}^\ell_{s,m}}[\|\mathcal{M}\|]$ be the expected number of distinct items in a Zipfian sample of size ℓ, and let $S_{s,m}(x) = \operatorname{argmin}_\ell\{D_{s,m}(\ell) \ge x\}$ be the expected number of samples needed to see x distinct items.

Lemma 1. *For any $m > 0$ and any constant $s > 0$, if $x = o(m)$, then $S_{s,m}(x) = \Theta(x^{\max\{1,s\}})$.*

Proof. Call the ith most probable item in the Zipfian a_i. Item a_i occurs $\ell\mathcal{Z}_{s,m}(a_i) = \ell/(i^s H_{s,m})$ times in $M \sim \mathcal{Z}^\ell_{s,m}$. Let v be the rank of the least frequent item that has expectation at least 1 in a sequence of $S_{s,m}(x)$ Zipfian samples, that is,

$$v = \max_i\left\{\frac{S_{s,m}(x)}{i^s H_{s,m}} \ge 1\right\}.$$

Notice that $(v+1)^s H_{s,m} > S_{s,m}(x) \ge v^s H_{s,m}$. Further, $v \le x$, and any item a_w where $w > v$ that occurs in our sample is expected to be unique. Thus $S_{s,m}(x)$ is the sum of the number of times a_1 through a_v appear in the sample, plus $\Theta(x-v)$, since the remaining items are all unique in expectation.

The expected number of times we sample the first v items is

$$\sum_{i=1}^{v}\frac{S_{s,m}(x)}{i^s H_{s,m}} = \frac{S_{s,m}(x)H_{s,v}}{H_{s,m}} \in [v^s H_{s,v}, (v+1)^s H_{s,v}).$$

The number of times we sample items a_w, where $w > v$, is the number of samples we haven't used up in frequent items, which is at most $S_{s,m}(x) - v^s H_{s,v} < (v+1)^s H_{s,m} - v^s H_{s,v}$. Thus, the number of distinct items found in $S_{s,m}(x)$ samples is v, plus the number of items with unique occurrences. In other words,

$$x \le v + (v+1)^s H_{s,m} - v^s H_{s,v}.$$

We now have a case analysis that depends on s. Note that for all s, $S_{s,m}(x) \ge x$ and that $S_{s,m}(x)$ is monitone with s, that is, if $s_1 < s_2$, then $S_{s_1,m}(x) \le S_{s_2,m}(x)$, which just means that as the distribution gets more concentrated, it takes more samples to get x distinct items.

- $s = 1$: For all y, $H_{1,y} \approx \ln y$. Therefore $x \le v + (v+1)\ln m - v\ln v = \Theta(v\ln m) = \Theta(v^1 H_{1,m}) = \Theta(S_{1,m}(x))$.
- $s < 1$: Lemma holds for $s = 1$ and by monotonicity.
- $s > 1$: $x \le v + (v+1)^s H_{s,m} - v^s H_{s,v} \le v + (v+1)^s \cdot \frac{v^{1-s}}{s-1} = \Theta(v)$. So we expect at least a constant fraction of elements that appear in the sample to appear multiple times. Such items appear $\Theta(v^s H_{s,v}) = \Theta(v^s) = \Theta(x^s)$ times.

□

3 Zipfian Adaptation

We analyze the expected behavior of broom filters and CAFs on Zipfian sequences.

We briefly summarize the aspects of the broom filter that are important for the purposes of this paper. First of all, a broom filter is a single-hash-function filter, which means that for each $x \in \mathcal{S}$, it maintains a hash $h(x)$ space-efficiently. (The default is that $h(x)$ hashes to a string of $\log n + \log(1/\varepsilon)$ bits, but as we will see, sometimes the hash function produces longer strings.) A query $y \in \overline{\mathcal{S}}$ is a false positive if $\exists x \in \mathcal{S}$ such that $h(x) = h(y)$. The broom filter adapts to false positives by storing more bits from x's hash value whenever an element $x \in \mathcal{S}$ is involved in a hash collision. These extra bits are called ***adaptivity bits***. Thus, if $y \in \overline{\mathcal{S}}$ is identified as a false positive because it collides with $x \in \mathcal{S}$, then the broom filter adds an expected 2 bits to $h(x)$ until the collision goes away.

In order to avoid the accretion of adaptivity bits to the hash function, the broom filter proceeds in ***rounds***: the filter

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

expires each hash function after it has been involved in $\Theta(n)$ false positives and replaces it with a new hash function. When a hash function expires, the adaptivity bits associated with the hash function are reclaimed. Thus, using only $\Theta(n)$ adaptivity bits, the broom filter can correct $\Theta(n)$ distinct false positives in the hash function, and $\Theta(n)$ false positives later, there is a new round with a new hash function and $\Theta(n)$ new adaptivity bits.

Theorem 2. *Let B be a broom filter on a set $\mathcal{S} \subset \mathcal{U}$, where $|\mathcal{S}| = n$, $|\mathcal{U}| = u$, $|\bar{\mathcal{S}}| = m = u - n$, and $n = o(u)$. If B has false-positive probability ε, then, in expectation, B's distributional false-positive rate on $\mathcal{Z}_{s,m}$ is at most $\min\left\{\varepsilon, \Theta\left(\frac{\varepsilon^s}{n^{s-1}}\right)\right\}$, for any constant $s > 0$.*

Proof. A round has $\Theta(n)$ false positives w.h.p., and each negative element in a round is a false positive at most once [4]. B begins each round with false positive probability ε which strictly decreases as B adds adaptivity bits. Therefore in expectation a round contains at least $\sum_{i=1}^{\Theta(n)} 1/\varepsilon = \Theta(n/\varepsilon)$ distinct negative elements.

By Lemma 1, if $0 < s \leq 1$, then in expectation $\Theta(n/\varepsilon)$ queries are required to observe $\Theta(n/\varepsilon)$ false positives, and so the false-positive rate is at most ε. Similarly, by Lemma 1, if $s > 1$, then seeing $\Theta(n/\varepsilon)$ distinct negative elements takes $\Theta((n/\varepsilon)^s)$ queries which yields a false-positive rate at most $\Theta(\varepsilon^s/n^{s-1})$.

□

In the remainder of this section we prove that in expectation the distributional false-positive rate of a broom filter with $\Theta(n)$ adaptivity bits on a Zipfian distribution $\mathcal{Z}$ is less than the lower bound of the distributional false-positive rate of a CAF with a $\Theta(n)$-bit cache on $\mathcal{Z}$.

For any function $\mathcal{X} : \mathcal{A} \to \mathbb{R}$, let $\text{rank}_{\mathcal{X},A}(a) = |\mathcal{L}| + 1$ where $\mathcal{L} = \{b \,|\, b \in \mathcal{A}, \mathcal{X}(b) < \mathcal{X}(a)\}$.

Lemma 3. *Let $F = (\mathcal{F}, \Delta)$ be a (non-adaptive) filter on $\mathcal{S}$, and let $\mathcal{X}$ be a distribution on $\overline{\mathcal{S}}$. For any $f \in \mathcal{F}$, let $\mathcal{M}_f = \{p_1, p_2, \ldots\}$ be the set of false positives for f, sorted in decreasing order of probability in $\mathcal{X}$. Then*

$$\mathop{\mathbb{E}}_{f\sim\Delta}[\text{rank}_{\mathcal{X},\bar{\mathcal{S}}}(p_i)] = \frac{i}{\varepsilon}.$$

Proof. By linearity of expectation. □

The caching problem for CAFs is not exactly the same as the traditional caching problem [25]. In the traditional caching problem, a page needs to be in the cache to be served, that is if the page is not in the cache, it is fetched into the cache to be served. For a CAF, if a false-positive item is not in the cache when it is queried, it is the filter's choice whether to store the false-positive item in the cache, evicting an existing false-positive to make room if the cache is full.

Consider a cache for which page requests are independently sampled from some distribution. The ***TOP-k*** cache policy keeps the k requests with the highest observed frequencies in cache.

Lemma 4. *Let C be a cache-augmented filter (CAF) with a cache of size k. The TOP-k cache policy minimizes the distributional false positive rate of C for any distribution on $\mathcal{S}$.*

Proof. C has non-adaptive filter $F = (\mathcal{F}, \Delta)$ and a cache of size k. Let f be a filter instance chosen randomly from Δ. Let $\mathcal{X}$ be a distribution on $\overline{\mathcal{S}}$, the set of negative queries, and let $\mathcal{X}^\ell$ be the probability distribution of strings of length ℓ where each character is drawn independently from $\mathcal{X}$. CAF C returns a false positive on a negative query if $f(x) = 1$ and $x \notin M$ where M is the set of false positives in the cache of CAF C at the time of the query. For simplicity, assume that at the beginning of the execution, the caching-policy loads a set M_0 of k false-positives into the cache. This can easily be achieved even if the cache starts empty and then loads false-positives as it progresses in the query sequence. The following function minimizes the distributional false-positive rate of C on $\mathcal{X}$:

$$\max_{\ell>0} \frac{C_\ell(M_0)}{\ell},$$

where $C_\ell(M_0)$ minimizes the number of false-positives of C on a sequence of length ℓ drawn independently from $\mathcal{X}$.

We recursively define the function $C_\ell(M_0)$ that minimizes the number of false-positives of a sequence of length ℓ drawn independently from $\mathcal{X}$, as follows. Given a negative query x, if x is a true negative (i.e. $f(x) = 0$) or x is in the cache, CAF does nothing and proceeds to the next query. Otherwise, x becomes a false-positive. The CAF then either chooses to keep the cache content the same for the next query or fetch x in cache and evict a false-positive y such that $C_{\ell-1}$ is minimized.

If $f(x) = 0$ or $x \in M_0$,

$$C_\ell(M_0) = \mathop{\mathbb{E}}_{x\in\bar{\mathcal{S}}}[C_{\ell-1}(M_0)]\,.$$

Otherwise,

$$C_\ell(M_0) = \mathop{\mathbb{E}}_{x\in\bar{\mathcal{S}}}[\alpha]\,.$$

where

$$\alpha = 1 + \min\left\{\begin{array}{l} C_{\ell-1}(M_0), \\ \min_{y\in M_0}\{C_{\ell-1}(M_0 \cup \{x\} \setminus \{y\})\} \end{array}\right\}.$$

We conclude the proof by noting that for two negative queries x and y such that $\Pr[x] > \Pr[y]$, $C_\ell(M \cup \{x\}) \leq C_\ell(M \cup \{y\})$. The proof of this claim is very similar to Lemma 2 in [2], which establishes the lemma. □

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

We next prove a lower bound on the distributional false-positive rate of a CAF on a Zipfian sequence. Together with Theorem 2, these results imply Theorem 6, analytically establishing the advantage of adaptivity over caching.

Theorem 5. *Let C be a CAF with a cache of size k and a false-positive probability of ε. The distributional false-positive rate of C on $Z_{s,m}$ is at least*

$$\varepsilon\left(1-\frac{H_{s,k/\varepsilon}}{H_{s,m}}\right).$$

Proof. By Lemma 4, we can assume that it uses the Top-k caching strategy. By Lemma 3, no query of up to an expected rank of k/ε causes a false positive on C. The probability that any particular query has rank up to k/ε is $H_{s,k/\varepsilon}/H_{s,m}$. The false positive rate on all other queries, which have an aggregate probability of $1-H_{s,k/\varepsilon}/H_{s,m}$, is ε which concludes the theorem. □

Finally, we compare the distributional false-positive rate of the broom filter with that of the CAF on $Z_{s,m}$.

Theorem 6. *Let B be a broom filter with cn adaptivity bits and a false-positive probability of ε, and let r_B be the distributional false-positive rate of B on $Z_{s,m}$. Let C be a CAF with a cn-bit cache and a false-positive probability of ε, and let r_C be the distributional false-positive rate of C on $Z_{s,m}$. $\mathbb{E}[r_B] \le \mathbb{E}[r_C]$ for any $s > 0$.*

Proof. By Theorem 2, $\mathbb{E}[r_B] \le \min\left\{\varepsilon, \Theta\left(\frac{\varepsilon^s}{n^{s-1}}\right)\right\}$. By Theorem 5, $\mathbb{E}[r_C] \ge \varepsilon\left(1-\frac{H_{s,k/\varepsilon}}{H_{s,m}}\right)$, where $k = n/\log u$.

When $s > 1$:

$$\begin{aligned}\mathbb{E}[r_C] &\ge \varepsilon\left(1-\frac{H_{k/\varepsilon,s}}{H_{m,s}}\right) \approx \varepsilon\left(1-\frac{1-(k/\varepsilon)^{1-s}}{1-m^{1-s}}\right)\\ &= \varepsilon\left(\frac{m^{s-1}-(k/\varepsilon)^{s-1}}{(km/\varepsilon)^{s-1}-(k/\varepsilon)^{s-1}}\right)\\ &\approx \varepsilon\left(\frac{m^{s-1}}{(km/\varepsilon)^{s-1}}\right) = \frac{\varepsilon^s}{k^{s-1}}\\ &> \Theta\left(\frac{\varepsilon^s}{n^{s-1}}\right) \ge \mathbb{E}[r_B]\end{aligned}$$

When $s \ll 1$,

$$\begin{aligned}\mathbb{E}[r_C] &\ge \varepsilon\left(1-\frac{H_{k/\varepsilon,s}}{H_{m,s}}\right) \approx \varepsilon\left(1-\frac{k}{\varepsilon m}\right)\\ &= \Theta(\varepsilon) \ge \mathbb{E}[r_B].\end{aligned}$$

The theorem follows from monotonicity. □

4 Experiment

In this section, we experimentally measure the performance of broom filters against that of CAFs and non-adaptive quotient filters. We show that broom filters offer a 18-74% reduction in false positive rate on simulated Zipfian data over CAFs and an even larger improvement over quotient filters. We also measure the performance of these filters on the network traces from [1]. For network data, broom filters continue to offer a 31-73% improvement over CAFs and more than 90% improvement over quotient filters.

Dataset	Number of unique items	Number of queries	Zipfian constant
Chicago A64	605006	14801266	1.19
Chicago B64	1700741	45359990	1.27
Sanjose 64	2193052	37815122	1.19

Table 1: CAIDA network trace datasets.

In each experiment, we store a set $\mathcal{S}$ in each filter, and then query elements (not in $\mathcal{S}$) from the data set to each filter, adapting when required. We characterize the empirical performance of a filter by the ratio of the number of false positives to the number of queries on negative items . We call this the ***empirical false positive rate (EFPR)***.

We use the quotient filter as our baseline static filter because it reliably offers the advertised false positive rate of ε on all types of input, and it is faster than alternatives such as the cuckoo filter.

4.1 Generated Zipfian Distribution Data. In order to generate Zipfian data, we set $\mathcal{U} = \{0, 1, 2, ..., \lfloor 80^{4.1} \rfloor\}$ and $\mathcal{S} = \{0, 1, 2, ..., 79\}$ so $n = 80$. We choose the value $n^{4.1}$ so that $\lfloor n/\log|\mathcal{U}| \rfloor = 3$, ensuring the caches for our CAFs can store some values. In many settings, $|\mathcal{U}|$ is larger than $80^{4.1}$ and a cache might not be able to store any items.

We repeated the experiment for six values of the Zipfian constant s: 0.05, 0.5, 0.9, 1, 1.1 and 1.2. For larger s, we observe that the broom filter's false positive rate drops to 0, though in theory a long enough query sequence could generate false positives.

The broom filter is allowed no more than $3n$ adaptive bits. The CAF-3/6/9 is equipped with a $n/2n/3n$-bit cache respectively, able to hold 3/6/9 items.The CAFs adopt the least-recently-used cache-eviction policy. We compare the three types of filters at several values of ε: 2^i for all values of $i \in [-10, -6]$. As a representative sample of the full results, the experimental findings for $s = 0.5$ and $s = 1.2$ are summarized in Figures 1 and 2.

These experimental findings validate our analysis: the broom filter outperforms the competition on Zipfian queries. In all cases the non-adaptive quotient filter has the highest EFPR, which is close to the false-positive probability (labeled as "ε"). The CAFs have basically the same EFPR as the non-adaptive quotient filter when $s \ll 1$. When $s > 1$, the CAFs have 52-91% lower EFPR than the non-adaptive

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

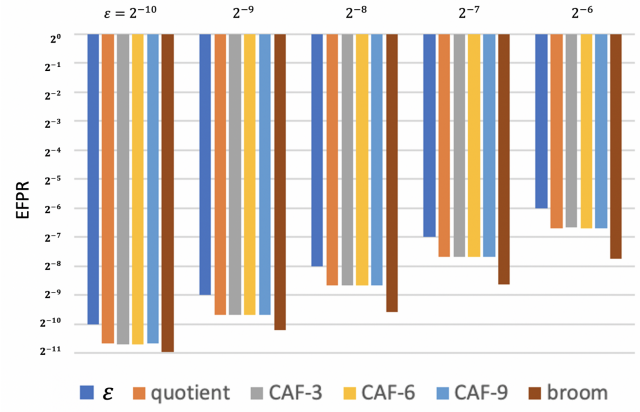

Figure 1: The EFPR of non-adaptive quotient filters, CAFs and broom filters with different false-positive probabilities ε on generated query sequences with Zipfian constant $s = 0.5$.

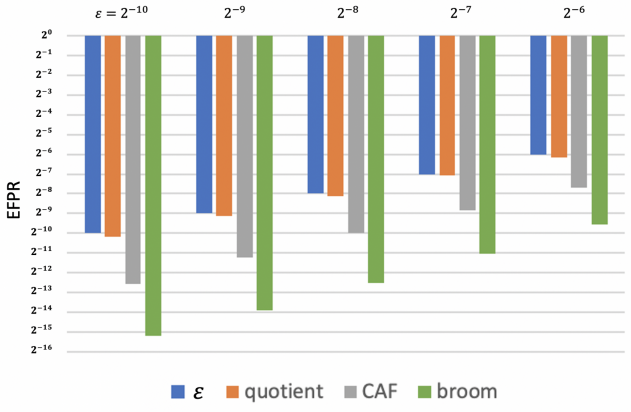

Figure 3: The EFPR of non-adaptive quotient filters, CAFs and broom filters with different false-positive probabilities ε on the Chicago A64 dataset.

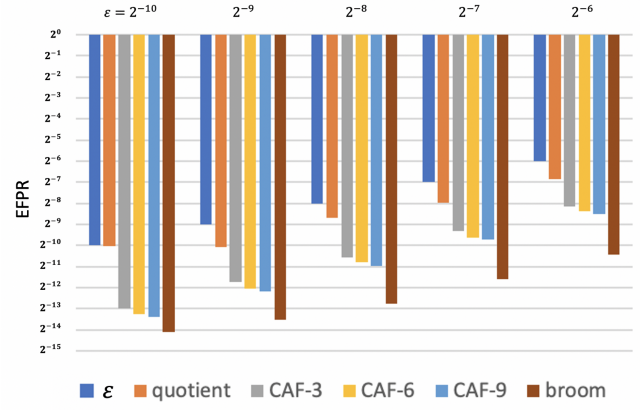

Figure 2: The EFPR of non-adaptive quotient filters, CAFs and broom filters with different false-positive probabilities ε on generated query sequences with Zipfian constant $s = 1.2$.

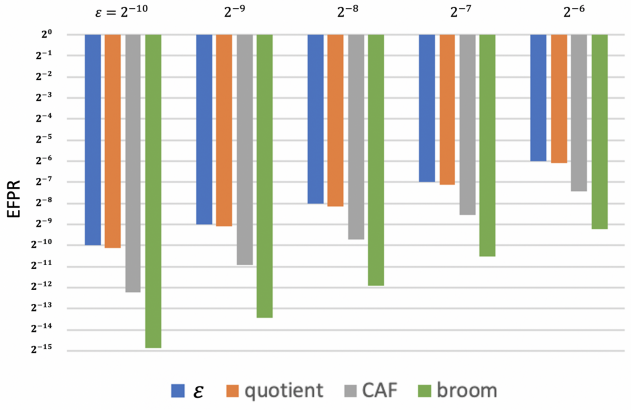

Figure 4: The EFPR of non-adaptive quotient filters, CAFs and broom filters with different false-positive probabilities ε on the Chicago B64 dataset.

quotient filter. All CAFs have higher EFPR than the theoretical result given by Theorem 5. The broom filter always achieves the lowest EFPR, offering a 18-74% reduction in EFPR over CAFs and 19-95% reduction over quotient filters. The EFPR of the broom filter is lower than the theoretical result given by Theorem 2.

4.2 Internet Trace Data. We measure the false positive rate achieved by each filter on network trace datasets. We use three such datasets from [1], whose characteristics are shown in Table 1. Note that we use Clauset, Shalizi, and Newman's method [11] to estimate the Zipfian constant of each dataset.

All items in the datasets are interpreted as negative queries. We generate a pseudo-positive set of size 30, because 30 is approximately the fourth root of our smallest negative set, i.e., the Chicago A64 set.

The broom filter is allowed no more than $3n = 90$ adaptive bits. The CAF is equipped with a 90-bit cache, and due to the large number of distinct items in these datasets this is only sufficient to cache 4 items. We compare the three types of filters at several values of ε: 2^i for all values of $i \in [-10, -6]$. The results are shown in Figures 3-5.

Once again, these experimental findings support our analytic results: the broom filter outperforms CAFs and quotient filters. The non-adaptive quotient filter has the highest EFPR, which is close to the false-positive probability (labeled as "ε"). Strikingly, the CAFs have 52-82% lower EFPR than the non-adaptive quotient filter despite the fact that the cache can hold no more than 4 items. All CAFs have higher EFPR than the theoretical result given by Theorem 5. The broom filter always achieves the lowest EFPR, offering 31-73% reduction in EFPR over CAFs and 90-97% reduction over quotient filters. The EFPR of the broom filter is lower

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

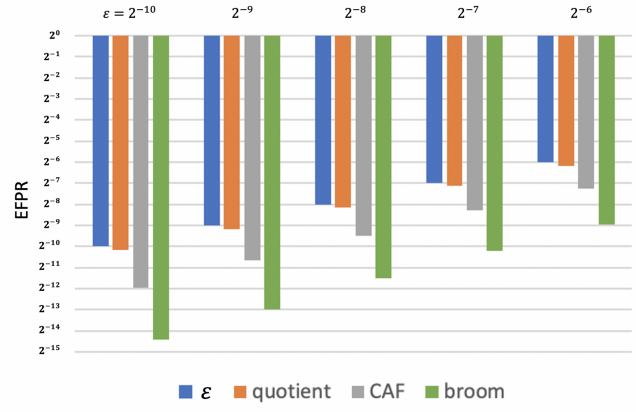

Figure 5: The EFPR of non-adaptive quotient filters, CAFs and broom filters with different false-positive probabilities ε on the San Jose 64 dataset.

than the theoretical result given by Theorem 2.

4.3 Serendipitous Corrections. A broom filter adapts in response to a false positive by extending the fingerprint of the element that collides with it. This may have the side effect of correcting other false positives that have not yet been presented to the filter. For example, if $\exists y, y' \in \overline{S}$ and $x \in S$ s.t. $h(y) = h(y') = h(x)$, then if the filter receives query y it will return a false positive and adapt, increasing the length of the x's fingerprint so that $h(y) \neq h(x)$. As a result, $h(y')$ may no longer be equal to $h(x)$, in which case if the filter receives query y' it will have avoided the false positive for no additional cost. We call the correction of any such y' ***serendipitous*** and call the correction of any such y ***direct***.

ε (FP Probability)	2^{-10}	2^{-9}	2^{-8}	2^{-7}	2^{-6}
Average Direct Corrections / round	1017	677	514	374	197
Avg. Serendipitous Corrections / round	403	421	361	248	206

Table 2: A comparison of the number of direct and serendipitous corrections observed for a broom filter on the Chicago A64 dataset. We report the average number of corrections per round for several values of ε.

We empirically investigate the prevalence of serendipitous corrections made by a broom filter in the network trace experiments described above. Table 2 summarizes the false positives corrected by an adaptive filter on the Chicago A64 dataset, and categorizes these corrections as serendipitous or direct. Strikingly, 24-55% of all observed corrections are serendipitous. The proportion of serendipitous corrections increases as false-positive probability ε increases, which is merely an artifact of our amortized implementation

ε (FP Probability)	2^{-10}	2^{-9}	2^{-8}	2^{-7}	2^{-6}
Average Direct Corrections / round	908	570	352	186	172
Avg. Serendipitous Corrections / round	301	282	255	206	198

Table 3: A comparison of the number of direct and serendipitous corrections observed for a broom filter on the Chicago B64 dataset. We report the average number of corrections per round for several values of ε.

ε (FP Probability)	2^{-10}	2^{-9}	2^{-8}	2^{-7}	2^{-6}
Average Direct Corrections / round	454	323	198	188	107
Avg. Serendipitous Corrections / round	222	228	182	190	134

Table 4: A comparison of the number of direct and serendipitous corrections observed for a broom filter on the San Jose 64 dataset. We report the average number of corrections per round for several values of ε.

of the broom filter (in contrast to the deamortized construction in [4]). Similar trends can be observed in the other network trace datasets in Tables 3 and 4. This phenomenon is a significant factor in the broom filter's excellent empirical false positive rate.

Acknowledgments

This research was supported in part by NSF grants CCF-1725543, CSR-1763680, CCF-1716252, CCF-1617618, CNS-1938709, CCF-1715777, AitF-1637458.

References

[1] The CAIDA UCSD Anonymized Internet Traces - 2019. `http://www.caida.org/data/passive/passive_dataset.xml`.

[2] A. V. Aho, P. J. Denning, and J. D. Ullman. Principles of optimal page replacement. *Journal of the ACM (JACM)*, 18(1):80–93, 1971.

[3] Y. Arbitman, M. Naor, and G. Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *Proc. IEEE 51th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 787–796, 2010.

[4] M. A. Bender, M. Farach-Colton, M. Goswami, R. Johnson, S. McCauley, and S. Singh. Bloom filters, adaptivity, and the dictionary problem. In *Proc. IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 182–193, 2018.

[5] M. A. Bender, M. Farach-Colton, R. Johnson, R. Kraner, B. C. Kuszmaul, D. Medjedovic, P. Montes, P. Shetty, R. P. Spillane, and E. Zadok. Don't thrash: how to cache your hash on flash. *Proc. VLDB Endowment*, 5(11):1627–1637, 2012.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

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

[7] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[8] J. Bruck, J. Gao, and A. Jiang. Weighted Bloom filter. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2304–2308, 2006.

[9] L. Carter, R. Floyd, J. Gill, G. Markowsky, and M. Wegman. Exact and approximate membership testers. In *Symposium on Theory of Computing*, pages 59–65, 1978.

[10] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The Bloomier filter: an efficient data structure for static support lookup tables. In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 30–39, 2004.

[11] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[12] N. C. Ellis and M. B. O'Donnell. Statistical construction learning: Does a Zipfian problem space ensure robust language learning. *Statistical learning and language acquisition*, 265:304, 2012.

[13] M. Eriksson, S. M. H. Rahman, F. Fraile, and M. Sjöström. Efficient interactive multicast over DVB-T2 - utilizing dynamic sfns and parps. In *2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–7, 2013.

[14] T. Kopelowitz, S. McCauley, and E. Porat. Support optimality and adaptive cuckoo filters. submitted.

[15] A. Kulkarni and A. Seetharam. Exploiting correlations in request streams: A case for hybrid caching in cache networks. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 562–570, 2018.

[16] C. Kurumada, S. C. Meylan, and M. C. Frank. Zipfian word frequencies support statistical word segmentation. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[17] C. Kurumada, S. C. Meylan, and M. C. Frank. Zipfian frequency distributions facilitate word segmentation in context. *Cognition*, 127(3):439–453, 2013.

[18] S. Lovett and E. Porat. A lower bound for dynamic approximate membership data structures. In *Foundations of Computer Science*, pages 797–804, 2010.

[19] B. Z. Manaris, L. Pellicoro, G. Pothering, and H. Hodges. Investigating Esperanto's statistical proportions relative to other languages using neural networks and Zipf's law. In *Artificial Intelligence and Applications*, pages 102–108. IASTED/ACTA Press, 2006.

[20] M. Mitzenmacher, S. Pontarelli, and P. Reviriego. Adaptive cuckoo filters. In *Proc. Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 36–47, 2018.

[21] A. Pagh, R. Pagh, and S. S. Rao. An optimal Bloom filter replacement. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 823–829, 2005.

[22] P. Pandey, M. A. Bender, R. Johnson, and R. Patro. A general-purpose counting filter: Making every bit count. In *Proc. International Conference on Management of Data (ICMD)*, pages 775–787, 2017.

[23] D. M. W. Powers. Applications and explanations of Zipf's law. In *New Methods in Language Processing and Computational Natural Language Learning*, 1998.

[24] K. D. Schuler, P. A. Reeder, E. L. Newport, and R. N. Aslin. The effect of Zipfian frequency variations on category formation in adult artificial language learning. *Language Learning and Development*, 13(4):357–374, 2017.

[25] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, February 1985.

[26] D. H. Zanette. Zipf's law and the creation of musical context. *CoRR*, cs.CL/0406015, 2004.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited