Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master Thesis

Optimization of Compact Set Membership Representation for Distributed Computing

March 2005

Andreas Diener
andreas.diener@vis.ethz.ch

Professor:
Prof. Dr. Roger Wattenhofer, ETH Zurich

Supervisors:
Dr. Paul Hurley, IBM Research GmbH
Prof. Dr. Marcel Waldvogel, University of Konstanz

Acknowledgements

I am grateful to Marcel Waldvogel, Paul Hurley, and Roger Wattenhofer for enabling and supervising this thesis. They provided all the help, support, and time needed for successfully completing this work and were valued contact persons at all times.

Also, I would like to say thank you to Patrick Droz, Jeroen Massar, Sankaranarayanan Sundararajan, Claudia Weber, Verena Diener, and Ulrich Diener who supported me with their time and ideas as well as in many enriching discussions.

This thesis was carried out at

IBM Research GmbH, Zurich Research Laboratory

Abstract

Bloom filters are widely used although not always optimal. Notably in application areas where false negatives are bearable, other techniques can clearly be better. This master thesis shows that at least for a specific area in the parameter space Bloom filters are significantly outperformed even by trivial methods. We provide an analysis, show that many application areas where Bloom filters are deployed do not require the strong "no false negatives" policy, and also examine application specific issues in a distributed web caching scenario. We believe we have opened up a new perspective with which to approach the subject and show that the unconditional use of Bloom filters is questionable.

Contents

1 Introduction **1**
- 1.1 Related Work . . . 2
- 1.2 Chapter Overview . . . 3

2 Improved Coding **5**
- 2.1 Problem Definition . . . 5
- 2.2 Bloom Filters . . . 6
- 2.3 Linear Optimization . . . 6
  - 2.3.1 Motivation . . . 6
  - 2.3.2 Defining the Optimization Problem . . . 6
  - 2.3.3 Stochastic Models . . . 7
  - 2.3.4 Approach: Test Cases . . . 7
  - 2.3.5 Approach: Mathematically Solving the Problem . . . 8
- 2.4 Cropped Filter . . . 10
- 2.5 Cost of One-sided Error . . . 11
- 2.6 N-K Model . . . 13
- 2.7 Optimal Coding . . . 15
- 2.8 Hamming Distance and Hamming Distortion . . . 16
- 2.9 Comparison of Different Methods . . . 16
  - 2.9.1 Derivation of Distortion Formulas . . . 17
  - 2.9.2 Visualization of the Distortion . . . 21
- 2.10 Cropped Bloom Filter . . . 23
- 2.11 Rate vs. Document Space . . . 25
- 2.12 Distortion vs. ρ . . . 26
- 2.13 Distortion for Huge and Realistic Document Spaces . . . 28

3 Simulation and Application Specific Issues **31**
- 3.1 Simulation Setup . . . 31
  - 3.1.1 Correctness of the Lookup Results . . . 31
  - 3.1.2 Computation Time . . . 32
  - 3.1.3 Workload . . . 32
- 3.2 Course of Action . . . 32
- 3.3 Implementing Compact Set Representation . . . 33
- 3.4 Simulation Results . . . 34

4 Conclusions **39**
- 4.1 Future Work . . . 39
- 4.2 Personal Experiences . . . 40

A Mid Thesis Report **41**
- A.1 Achievements . . . 41
- A.2 Future Work . . . 42

B Summary **43**

C Zusammenfassung **45**

Chapter 1

Introduction

Bloom filters [Blo70] pop up everywhere in networking. In the last four years, the original article from 1970 was cited in more than 51 publications from the ACM alone. It's the tool of first and last resort when looking at how to compactly store set membership information in some form or another.
We posed the question as to whether better performance could be achieved; or if different applications necessitated, or would benefit from other schemes for set membership representation. This work shows that indeed Bloom may, in certain circumstances, be far from optimal, and even very simple schemes may do better and be more appropriate.
Many efforts have focused on enhancing Bloom filters, while one work [Mit01] showed the benefit of compressing them. We take this one stage further and return to the underlying goal - the compression of set membership information. This helps in understanding the limitations (and also advantages) of Bloom filters and aids in the derivation of other schemes that may be tailored to the particular networking application.
Set membership representation is a data structure that stores information as to whether an element is contained in a set or not. Compact set membership representations can significantly reduce the memory size necessary by trading off accuracy for storage cost. By far the most prominent method is the aforementioned Bloom filter. It, along with its derivatives and enhancements, is the sole method applied in many network applications to date.
Bloom filters were first used to circumvent shortage of local memory and were mainly applied in the area of databases. Today, they have regained attention in distributed computing where communication cost is one of the limiting factors. They now serve as both a method for storage as well as for set membership information exchange. Some application areas of Bloom filters are listed below.

Distributed Caching Does another cache contain a document? [FCAB00]

Object Location in P2P Networks Keep a Bloom filter for the objects of every other node in the network. [RK02]

Approximate Set Reconciliation Efficient calculation of $S_A - S_B$. [BCMR02]

Set Intersection Efficient finding of $S_A \cap S_B$. [BHPW04]

The restriction of only having false positives is a strong characteristic, which, intuitively, should come at a price. When this characteristic is required or helpful, this price may be worth paying. However, Bloom filters are so handy (and currently the only kid on the block), that they seem to be used independently of such considerations. Table 1 illustrates some of the most prominent application areas of Bloom

Application area [BM02]	**Necessity of one-sided error**
Distributed caching	No
Object location in P2P systems	No
Approximate set reconciliation	No*
Resource routing	No
Loop detection	No*
Flow detection	Yes
Multicast	Yes
Hyphenation exceptions	Yes
Set intersection	Yes
Differential files	Yes

* False negatives preferable

Table 1.1: *Necessity of restriction to only having false positives.*

filters and the corresponding necessity of only having a one-sided error. A more detailed description of these applications and their settings can be found in [BM02]. One application that does not rely on a one-sided error is distributed caching, where validity of resources may expire and at the same time new ones become available on the web. Here, the existence of both false positives and false negatives, while not particularly cherished, are tolerable. The use of a set membership representation restricted to one-sided error can be undesired.
In some applications, false negatives are even less costly than false positives. For example, consider the problem of calculating set subtraction as part of approximate set reconciliation. False positives are responsible for the faulty outcome, whereas false negatives would simply cause some overhead.
These reflections are motivation for an analysis of different methods of compact set membership representation. Thereafter, it is looked at how well the findings can be applied in practice and finally a performance evaluation of a distributed cache using different methods of set membership storage is carried out.

1.1 Related Work

During the past 35 years, the Bloom filter captivated many researchers and a multitude of evolutions subsequently emerged. Most go in the direction of new functionality, which is not the scope of this work. One example is the *counting Bloom filter*[FCAB00], which introduces the ability to remove elements. Instead of a bit array, the counting Bloom filter uses a small number of bits per entry to keep count of the number of elements, incremented upon insertion and decremented upon deletion. The *bloomier filter*[CKRT04] looks at a Bloom filter as a data structure for compactly encoding a function. It extends the existing filter, which can only handle set membership queries, in order to encode arbitrary functions. The Bloom filter also became an integral part of more complex methods and algorithms. The *attenuated Bloom filter*[RK02] for example uses Bloom filters for dynamic document location.
The *compressed Bloom filter* by Mitzenmacher [Mit01] goes in the direction of performance improvement, in the spirit of this work. It alters the Bloom filter in a way that improves the accuracy for a given size using compression. Mitzenmacher looks

at Bloom filters as both a data structure to be used at proxies and as a message to be passed between them. A new optimization problem is set up with the message size as a newly introduced parameter. The compression rate of a compressed Bloom filter depends on the size of the local Bloom filter and the number of hash functions used.
In the context of image compression, Weidmann and Vetterli [WV01, WV99] study spike processes and in particular the rate-distortion of sparse memoryless sources. This can also be applied to the study of set representation, as explained later in Chap. 2.
Two papers are associated with this thesis. [HDW05] investigates the general nature of one-sided errors and presents the subject of lossy set membership representation from an information theoretical viewpoint. [DHW05] focuses on practical aspects in network applications and examines the use of alternative methods to Bloom filters.

1.2 Chapter Overview

The task description of this thesis targets at optimizing search in distributed network storage and proposes hierarchical coding, improved coding, and exchange of relevance information as possible ways to do so. Since promising results were found in the area of improved coding, this master thesis focuses on this topic only.
The rest of this report is organized as follows.

Chapter 2, Improved Coding An analysis of existing and newly proposed coding schemes of compact set membership representation.

Chapter 3, Simulation and Application Specific Issues Performance evaluation of a distributed cache scenario using different methods of compact set membership representation.

Chapter 4, Conclusions Final conclusions summarizing the results and personal experiences.

Appendix A, Mid Thesis Report A short intermediary report handed in after three months.

Appendix B, Summary A summary of this thesis in English.

Appendix C, Zusammenfassung Eine Zusammenfassung dieser Arbeit in deutscher Sprache.

Chapter 2

Improved Coding

The chapter at hand documents an analysis of existing and new methods of compact set membership representation. Emerging from an initial linear optimization problem, the *cropped filter* is proposed as one trivial method that outperforms the well known Bloom filter in at least some areas of the parameter space. While investigating possible reasons for this unexpected finding, the cost of an error restriction to only false positives or only false negatives is discussed. A comparison of several set membership representations with respect to distortion is given and illustrated in several graphs.

2.1 Problem Definition

The problem of compact set membership representation is defined as follows. Consider a set of objects $G = \{g_1, g_2, g_3, ...\}$ and a subset $S \subseteq G$. A compact set membership representation of S is a data structure that stores information as to whether an element is contained in S or not. In some applications, infrequent wrong answers are bearable and the introduced space savings outweigh the disadvantage of a small error probability.
The two possible basic errors are:

False Positive If an element is wrongly determined to be in S.

False Negative If an element is wrongly determined not to be in S (i.e. in $G \setminus S$).

Compact set membership representations can have distinct characteristics. Bloom filters make enumeration of the represented subset difficult. This quality and the existence of false positives lead to some application scenarios:

Vote Controlling The social security numbers of the people who have already casted their vote are inserted into a Bloom filter (or similar compact set representation) which is then distributed to all polling stations. This Bloom filter allows one to verify if a person has already voted. In the case of a positive answer, a validation with a central instance storing the uncompressed list is necessary to prevent false positives. One advantage of such an approach is that it is not possible to enumerate the people who have voted using the Bloom filter alone, thus introducing a level of privacy.

Electronic Money The ID numbers of valid, unspent electronic money are available for download in form of a Bloom filter. The fairly high probability that false money is identified as such and the unpredictability thereof prevents abuse.

License Plate Scanning Police men carry a portable device containing a Bloom filter of suspicious license plate numbers. Space savings, the awareness of the existence of false positives, and the impossibility of an enumeration are advantages compared to storing the list directly.

2.2 Bloom Filters

A prominent method of compact set membership representation is the Bloom filter [Blo70] that was invented by Burton Bloom in 1970. It consists of an array of bits, initially all set to zero. To insert an element, several hash functions with range over the bitmap are calculated. The bits corresponding to the hash function output are set to one. Set membership is ascertained by performing an AND operation on all the bits stored at the indices that equal the hash values of the requested element. The property of the filter that stands out is the production of one-sided errors in membership determination. An inserted element will always be correctly ascertained as being a member of the set. Bloom filters do not produce any false negatives. The flip-side is that elements not in the set may falsely be determined as belonging to the set - a so-called false positive. This is because all hash functions could map to values that were previously set to one.

2.3 Linear Optimization

2.3.1 Motivation

Bloom filters are one method for lossy set membership representation. In search for alternative and possibly better methods we looked at the problem from a general and abstract viewpoint. We hoped to be able to profit from and incorporate findings in the area of general data compression. Bloom filters allow for false positives but no false negatives. If a method is allowed to contain both false positives and false negatives, it might be possible to fundamentally improve the compression rate.

2.3.2 Defining the Optimization Problem

We try to find an optimal method of compact set membership representation by solving an optimization problem. For simplicity, the first approach is carried out in $\mathbb{Z}_2$. The problem having an optimal method of compact set membership representation as solution is defined as follows. Consider the finite set of possible documents $G = \{g_1, ..., g_n\}$ with $|G| = n$. Let S be a subset of G. Let $x \in \mathbb{Z}_2^n$ with $g_s \in S$ iff $x_s = 1$, thus a vector containing the output of the characteristic function of S. Let $y \in \mathbb{Z}_2^m$ be a lossy representation of the set S. In general, $y = f(x)$ for some function f. Let the *recovery* of x be $\hat{x} = f^{-1}(y)$. Thus, y can be considered a lossy compression of x with its approximate recovery given by $\hat{x}$.
Consider the case where f and f^{-1} are linear and

$$Ax = y$$

and

$$By = \hat{x}.$$

A and B are $m \times n$ and $n \times m$ 0-1 matrices.
No false negatives exist if $x_s = 1 \Rightarrow \hat{x}_s = 1 \; \forall s \in S$.
No false positives exist if $\hat{x}_s = 1 \Rightarrow x_s = 1 \; \forall s \in S$.
The number of mistakes is the Hamming distance between x and $\hat{x}$.

$$d(x, \hat{x}) = x^T \hat{x}$$

2.3.3 Stochastic Models

In this optimization problem the document space is restricted to be finite. Two basic probabilistic models for constructing the subset that is to be represented are the following.

Bernoulli Model $\Pr(x \in S) = \rho$. Let

$$x = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$

with X_i as Bernoulli random variable i.e. $\Pr(X_i = 1) = \rho$.

N-K Model $|S| = k$ known and all set membership combinations are equiprobable – i.e. permutations with k 1's in n positions are possible.

We start off with the Bernoulli model, later look at the N-K model and analyze the differences.

2.3.4 Approach: Test Cases

In order to get a feeling for the problem, some exemplary test cases were solved. This was done by a brute force approach using the Matlab function below.

```
function [] = linoptimize(n,m)
% optimizes over all linear lossy set representations
% finds the transformation matrices A, B that minimize
the distortion
% size of document space = n
% size of compressed vector y = m

% initialize minDist to the maximum possible value
minDist = n; minA = 0; minB = 0;

% try all possible A matrices
for i = 0:1:((2^(n*m))-1)
        % try all possible B matrices
        for j = 0:1:((2^(n*m))-1)
                A = dec2binmat(i,n,m);
                B = dec2binmat(j,m,n);
                avg = getaveragedistortion(A,B,n);
                if avg < minDist
                        minDist = avg; minA = A, minB = B
           end
        end
end
```

Unfortunately, the above algorithm proves to be quite inefficient, i.e. for values $n = 5$ and $m = 4$ execution time exceeds 20 hours! But it correctly solves the small examples and each produces lots of resulting matrices that optimize the problem. The optimization for values $n = 3$ and $m = 2$ produces 18 results. They include following matrices A and B:

$$dist = 0.1 \qquad A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$dist = 0.1 \qquad A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$dist = 0.1 \qquad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$dist = 0.1 \qquad A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$dist = 0.1 \qquad A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

These results seem to obey certain patterns that lead to the following hypotheses:

- The identity matrix filled up with zeros is part of the result set.
- All matrices A that result from column permutations of a matrix A that minimizes the Hamming distance combined with the appropriate matrix B also minimize the Hamming distance.
- $A * B = I$.
- All optimal matrices have full rank.

2.3.5 Approach: Mathematically Solving the Problem

The linear optimization problem is, given x, determined by matrices A and B such that the Hamming distance between x and $\hat{x}$ is minimized. In this section we mathematically derive the linear transform that minimizes this Hamming distance. In other words, we determine A and B such that $\mathbf{E}[d(x,\hat{x})] = \mathbf{E}[d(x, BAx)]$ is minimized. In this instance, the problem is limited to having all elements of A and B in $\mathbb{Z}_2$ (i.e. 0-1 matrices).
First, we look at $\mathbf{E}[X \oplus Y]$, for X and Y random variables.

$$\begin{aligned}
\mathbf{E}[X \oplus Y] &= \sum_{x\in\{0,1\}} \sum_{y\in\{0,1\}} x \oplus y \Pr(X = x, Y = y) \\
&= \sum_{(x,y)\in\{(0,1),(1,0)\}} x \oplus y \Pr(X = x, Y = y) \\
&= \Pr(X = 0, Y = 1) + \Pr(X = 1, Y = 0)
\end{aligned}$$

which also is intuitively what one would expect.
Each output Y_i from the filter is

$$Y_i = \sum_{j=1}^{n} a_{ij} X_j .$$

Then

$$\begin{aligned}\hat{X}_i &= \sum_{k=1}^{m} b_{ik} Y_k \\ &= \sum_{k=1}^{m}\sum_{j=1}^{n} b_{ik} a_{kj} X_j.\end{aligned}$$

Since expectation is linear, we can write

$$\begin{aligned}\mathbf{E}[d(x,\hat{x})] &= \sum_{i=1}^{n} \mathbf{E}[X_i \oplus \hat{X}_i] \\ &= \sum_{i=1}^{n} \Pr(X_i = 0, \hat{X}_i = 1) + \Pr(X_i = 1, \hat{X}_i = 0) \\ &= \sum_{i=1}^{n} \Pr(X_i = 0, 1 = \sum_{k=1}^{m}\sum_{j=1}^{n} b_{ik} a_{kj} X_j) \\ &\quad + \Pr(X_i = 1, 0 = \sum_{k=1}^{m}\sum_{j=1}^{n} b_{ik} a_{kj} X_j).\end{aligned} \tag{2.1}$$

We consider the case $n = 3$ and $m = 2$.
For $i = 1$ we get

$$\Pr(X_1 = 0, 1 = \sum_{k=1}^{2}\sum_{j=1}^{3} b_{1k} a_{kj} X_j) + \Pr(X_1 = 1, 0 = \sum_{k=1}^{2}\sum_{j=1}^{3} b_{1k} a_{kj} X_j). \tag{2.2}$$

We analyze the 'double sum' of the first term and get

$$\begin{aligned}&\sum_{k=1}^{2}\sum_{j=1}^{3} b_{1k} a_{kj} X_j = \\ &\quad = b_{11}a_{11}X_1 + b_{12}a_{21}X_1 + b_{11}a_{12}X_2 + b_{12}a_{22}X_2 + b_{11}a_{13}X_3 + b_{12}a_{13}X_3 \\ &\quad = X_1(b_{11}a_{11} + b_{12}a_{21}) + X_2(b_{11}a_{12} + b_{12}a_{22}) + X_3(b_{11}a_{13} + b_{12}a_{23}) \\ &\quad = X_2\alpha_1 + X_3\beta_1 + X_1\gamma_1\end{aligned}$$

where

$$\begin{aligned}\alpha_1 &= b_{11}a_{12} + b_{12}a_{22} \\ \beta_1 &= b_{11}a_{13} + b_{12}a_{23} \\ \gamma_1 &= b_{11}a_{11} + b_{12}a_{21}.\end{aligned}$$

For the Bernoulli random variable X_i we derive the probabilities

$$\begin{aligned}\Pr(X_1 = 0) &= 1 - \rho \\ \Pr(X_i\alpha_1 = 0) &= 1 - \rho\alpha_1 \\ \Pr(X_i\alpha_1 = 1) &= \rho\alpha_1.\end{aligned} \tag{2.3}$$

Going back to (2.2) and applying Baye's rule

$$\Pr(AB) = \Pr(A \mid B) * \Pr(B)$$

as well as (2.3) we get

$$
\begin{aligned}
\Pr(X_1 = 0, 1 = \sum_{k=1}^{2}\sum_{j=1}^{3} b_{1k}a_{kj}X_j) &= \\
&= \Pr(X_1 = 0) * \Pr(1 = \sum_{k=1}^{2}\sum_{j=1}^{3} b_{1k}a_{kj}X_j) \mid X_1 = 0) \\
&= (1-\rho)\Pr(1 = X_2\alpha_1 + X_3\beta_1 + 0) \\
&= (1-\rho)(\Pr(X_2\alpha_1 = 0, X_3\beta_1 = 1) + \Pr(X_2\alpha_1 = 1, X_3\beta_1 = 0)) \\
&= (1-\rho)((1-\rho\alpha_1)\rho\beta_1 + \rho\alpha_1(1-\rho\beta_1)) \\
&= \rho(\alpha_1+\beta_1) - \rho^2(2\alpha_1\beta_1 + \alpha_1 + \beta_1) + \rho^3(2\alpha_1\beta_1).
\end{aligned}
$$

After repeating these steps for all terms of (2.1) we find the closed formula for the Hamming distance

$$
\mathbf{E}[X \oplus Y] = \sum_{i=1}^{3} \rho^3(2\alpha_i\beta_i - 4\alpha_i\beta_i\gamma_i) + \rho^2(2\alpha_i\gamma_i + 2\beta_i\gamma_i - \alpha_i - \beta_i) + \rho(1-\gamma_i)
$$

where

$$
\begin{aligned}
\alpha_1 &= b_{11}a_{12} + b_{12}a_{22} \\
\beta_1 &= b_{11}a_{13} + b_{12}a_{23} \\
\gamma_1 &= b_{11}a_{11} + b_{12}a_{21}. \\
\\
\alpha_2 &= b_{21}a_{11} + b_{22}a_{21} \\
\beta_2 &= b_{21}a_{13} + b_{22}a_{23} \\
\gamma_2 &= b_{21}a_{12} + b_{22}a_{22}. \\
\\
\alpha_3 &= b_{31}a_{11} + b_{32}a_{21} \\
\beta_3 &= b_{31}a_{12} + b_{32}a_{22} \\
\gamma_3 &= b_{31}a_{13} + b_{32}a_{23}.
\end{aligned}
$$

Minimizing this expression for all possible elements of A and B we get the same results for the case $r = 3$ and $m = 2$ as with the brute force approach discussed in Sect. 2.3.4 and therefore suspect this formula to hold without explicit proof.

2.4 Cropped Filter

Resulting from the hypothesis that the identity matrix filled up with zeroes minimizes the linear optimization problem, the cropped filter is proposed.

$$
y = Ax = \begin{bmatrix} & 0 & \cdots & 0 \\ I & \vdots & & \vdots \\ & 0 & \cdots & 0 \end{bmatrix} * \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix}
$$

Put in words, the compact representation of the document vector x is simply its first m bits.

The reconstruction of the cropped filter is then given by

$$\hat{x} = By = \begin{bmatrix} & I & \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} * \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ for } \rho \leq 0.5$$

and

$$\hat{x} = By = \begin{bmatrix} & I & \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} * \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \\ 1 \\ \vdots \\ 1 \end{bmatrix} \text{ for } \rho > 0.5.$$

$\hat{x}$ consists of the first m bits of x which are represented in the cropped filter y and a maximum likelihood decoding of the remaining positions that we do not have any further information about. For $\rho \leq 0.5$ they are set to 0, for $\rho > 0.5$ to 1.
When using cropped filters, the resulting compression is not very impressive, but one can combine this approach with an additional compression step by first using the linear optimal method, cropped filters, and thereafter applying a compressor in order to reduce the size of the transmitted message.
The following illustrates this process. The document space with size n is transformed to a intermediary lossy representation y of size q using a cropped filter. y is then compressed to $y^{'}$ which is of size m.

$$x \xrightarrow{A} y \xrightarrow{compress} y^{'}$$

We assume that we have an optimal compressor, that is, we assume that our cropped filter of size q can be compressed down to only $qH(\rho)$ bits, where

$$H(\rho) = -\rho \log_2 \rho - (1-\rho) \log_2 (1-\rho)$$

is the entropy function. This compressor therefore uses $H(\rho)$ bits on average for each bit in the original filter.
In practice, near-optimal compressors exist. Arithmetic coding for example requires on average less than $H(\rho) + \epsilon$ bits per character for any $\epsilon > 0$ given suitably large strings. Going from the cropped filter to the compressed cropped filter is a similar step than going from the Bloom filter to the compressed Bloom filter as described in [Mit01].

2.5 Cost of One-sided Error

This section addresses the inevitable price that has to be paid when using methods with a one-sided error. We derive the rate-distortion curve when no false negatives are permitted under the assumption of an input conforming to a Bernoulli process.

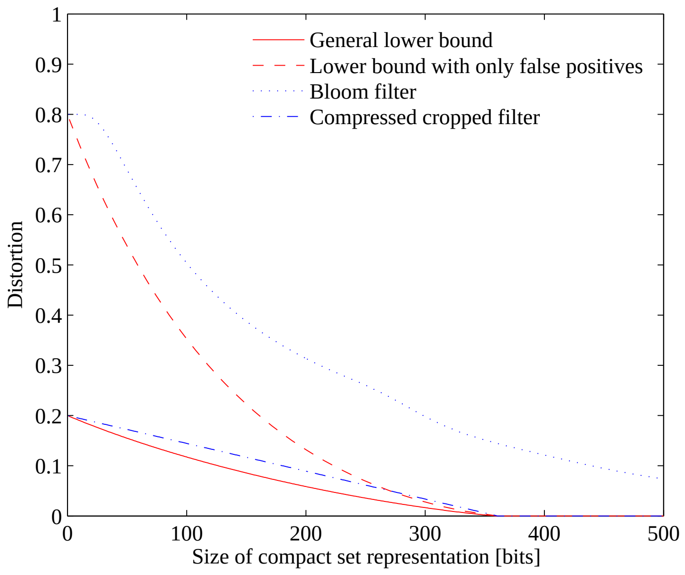

Figure 2.1: *Theoretical error bound and theoretical error bound for methods only having false positives.*

In order to prevent the existence of false negatives we define the distortion metric as follows.

$$d(x,\hat{x}) = \begin{cases} 0, & x = \hat{x}, \\ 1, & x = 0, \hat{x} = 1, \\ \infty & x = 1, \hat{x} = 0. \end{cases}$$

The input X is a Bernoulli-ρ random variable with $\rho \leq \frac{1}{2}$ and $\hat{X}$ is the output of any coder. We first derive $f(x,\hat{x}) = \Pr(X = x, \hat{X} = \hat{x})$, the joint distribution of X and $\hat{X}$. In order to have no false negatives and thus a finite distortion, $f(1,0) = 0$. $f(0,1)$ is the only remaining constellation contributing to the distortion and must therefore be equal to the entire distortion $f(0,1) = D$. Since the total probability $\Pr(X = 1) = \rho$, we know that $f(1,0) + f(1,1) = \rho$ and find $f(1,1) = \rho$. Since $\Pr(X = 0) = f(0,0) + f(0,1) = 1 - \rho$, we find $f(0,0) = 1 - \rho - D$. Thus, the joint distribution is

$$f(x,\hat{x}) = \begin{bmatrix} 1-\rho-D & D \\ 0 & \rho \end{bmatrix}.$$

Note that $\Pr(\hat{X} = 0) = 1 - \rho - D$ and $\Pr(\hat{X} = 1) = D + \rho$.

$$\begin{aligned} -H(X|\hat{X} = 0) &= \frac{f(0,0)}{\Pr(\hat{X} = 0)} \log_2 \frac{f(0,0)}{\Pr(\hat{X} = 0)} + \frac{f(1,0)}{\Pr(\hat{X} = 0)} \log_2 \frac{f(1,0)}{\Pr(\hat{X} = 0)} \\ &= 0 + 0 \log_2 0 = 0, \end{aligned}$$

taking the standard convention that $0 \log_2 0 = 0$, since this is true at the limit.

$$\begin{aligned} -H(X|\hat{X} = 1) &= \frac{f(0,1)}{\Pr(\hat{X} = 1)} \log_2 \frac{f(0,1)}{\Pr(\hat{X} = 1)} + \frac{f(1,1)}{\Pr(\hat{X} = 1)} \log_2 \frac{f(1,1)}{\Pr(\hat{X} = 1)} \\ &= \frac{D}{D+\rho} \log_2 \frac{D}{D+\rho} + \frac{\rho}{D+\rho} \log_2 \frac{\rho}{D+\rho}. \end{aligned}$$

Thus,

$$-H(X|\hat{X}) = D \log_2 \frac{D}{D+\rho} + \rho \log_2 \frac{\rho}{D+\rho}$$

and

$$\begin{aligned}
I(X;\hat{X}) &= H(X) - H(X|\hat{X}) \\
&= -\rho \log_2 \rho - (1-\rho)\log_2(1-\rho) + D \log_2 \frac{D}{D+\rho} + \rho \log_2 \frac{\rho}{D+\rho} \\
&= -(1-\rho)\log_2(1-\rho) + D\log_2 D - (D+\rho)\log_2(D+\rho)
\end{aligned}$$

from which the following theoretically reachable rate at a distortion consisting of only false positives results.

$$R(D) = \begin{cases} -(1-\rho)\log(1-\rho) + D\log D - (D+\rho)\log(D+\rho), & D \le 1-\rho, \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

In a similar manner we find the theoretically reachable rate at a distortion consisting of only false negatives to be

$$R(D) = \begin{cases} -\rho\log\rho + D\log D - (1-\rho+D)\log(1-\rho+D), & D \le \rho, \\ 0, & \text{otherwise.} \end{cases}$$

The lower bound of (2.4) for methods restricting possible errors to only false positives is plotted in Fig. 2.1. The lower bound for general methods without restriction on the error will be derived later in Sect. 2.9.1. One can clearly see that especially for small sizes of the compact set representation, more bits are necessary to reach a given distortion compared to when using methods with no restriction on error. One can also see that the Bloom filter is close to the lower bound for one-sided error methods and that the compressed cropped filter is close to the general lower bound. As a result of the significant difference of the two bounds, the cropped filter's performance is below the lower bound of methods restricted to a one-sided error which proves that for most sizes of the representation and an equiprobable set membership of $\rho = 0.2$, the compressed cropped filter performs better than any one-sided error method such as the Bloom or the compressed Bloom filter.
A more in-depth analysis of lossy set membership representation from an information theoretical viewpoint can be found in a companion paper to this report [HDW05].

2.6 N-K Model

In [WV99] and [WV01] Weidmann and Vetterli describe the rate-distortion behavior of sparse memoryless sources. The subsequent section summarizes their findings and relates their work to the problem of compact set membership representation.
The notion of *multiple spikes* corresponds exactly to the problem of set representation in the N-K model as described in Sect. 2.3.3. A multiple spike is a source that emits a vector of length N with K bits set to 1 – the so-called spikes – and the rest to 0. For given N and K there are $\binom{N}{K}$ such binary vectors. The rate-distortion function of such a source consists of a linear as well as a nonlinear part. The linear part is

$$R(D) = (D-K)\log_2 \beta_0, \qquad D(\beta_0) < D < K$$

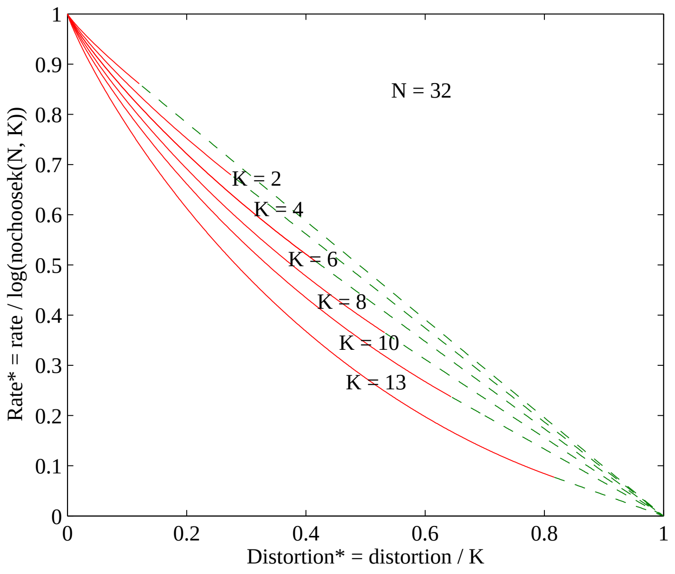

Figure 2.2: *Rate-distortion function in the N-K model for a fixed document space of $N = 32$ and $K = 2, 4, 6, 8, 10,$ and 13 elements. The curves contain a linear (dashed) and a non-linear (solid) segment.*

where β_0 is the solution to the equation

$$\sum_{d=0}^{K} w_d \beta^{2d} - \binom{N}{K} \beta^K = 0$$

where

$$w_d = \binom{K}{d}\binom{N-K}{d}, \qquad d = 0, \ldots, K.$$

The parametric expression for the information rate-distortion curve for $0 < \beta < \beta_0$ is given by

$$D(\beta) = \sum_{d=1}^{K} b_d 2d,$$

$$R(\beta) = \log_2 \binom{N}{K} + \sum_{d=0}^{K} b_d \log_2 b_d - \sum_{d=0}^{K} b_d \log_2 w_d$$

for

$$b_d = \frac{w_d \beta^{2d}}{\sum_{d'=0}^{K} w_{d'} \beta^{2d'}}, \qquad d = 0, \ldots, K.$$

Figure 2.2 shows the information rate-distortion curves for $N = 32$ and different K. One can see that for small K/N the linear segment dominates the rate-distortion behavior and in applications of set representation, K/N is typically very small. As a result of this close to linear behavior of $R(D)$, we can construct a near optimal encoding: encode one part of the spikes exactly with the necessary rate for a lossless encoding of $\log \binom{N}{K}$ and do not encode the rest at all using a rate of 0. This encoding results in a linear $R(D)$ behavior and it can be seen above, that for small K this

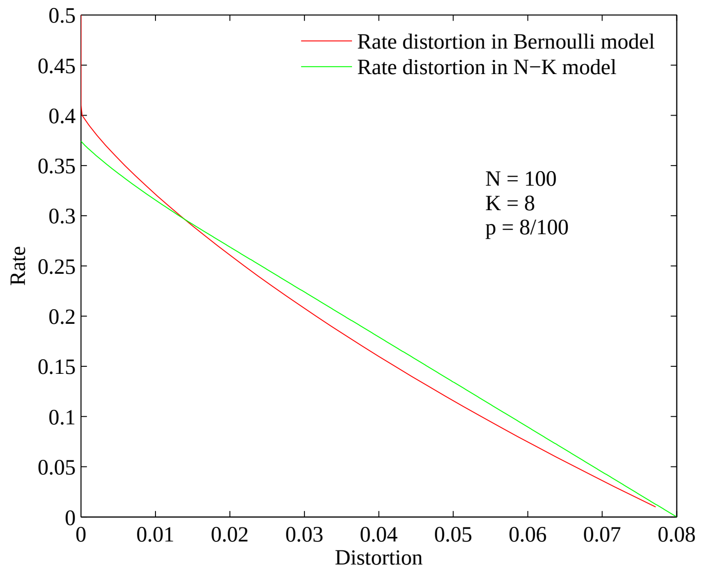

Figure 2.3: *Comparison of rate-distortion curves for the Bernoulli and the N-K model. The two curves are not the same but stay very close.*

is very close to the rate-distortion and therefore almost optimal. This encoding corresponds to the cropped filter in the N-K model.
The Bernoulli and the N-K model are in many ways similar but they still do reveal some fundamental differences as it is shown in the expressions for the information rate-distortion. In Fig. 2.3 the information rate-distortion curves in the N-K model as well as the Bernoulli model are shown for $N = 100$ and $K = 8$ or $\rho = 8/100$ respectively. In order to indicate that these rate-distortion curves also stay close together for realistic values of N, here the entropy for $N = 2^{32}$, $K = 10\,000$, and $\rho = 10\,000/2^{32}$:

$$\begin{aligned} H(\text{Bernoulli}) &= NH(\rho) \\ &= 201550 \\ H(\text{N-K model}) &= \binom{N}{K} H\left(\frac{1}{\binom{N}{K}}\right) \\ &= 201541 \end{aligned}$$

This difference is less than 50 ppm.

2.7 Optimal Coding

It can be shown for the Bernoulli model with small ρ, that the compressed cropped filter is asymptotically optimal. Namely, that the normalized rate-distortion function is asymptotically linear as ρ goes to 0 [WV01, WV99]. The normalized rate-distortion function for the normalized Hamming distortion $d = \frac{D}{\rho}$ is given by

$$\begin{aligned}\frac{R(d)}{H(\rho)} &= 1 - \frac{H(\rho d)}{H(\rho)} \\ &= 1 - \frac{\rho d \log(\rho d) + (1-\rho d)\log(1-\rho d)}{\rho \log(\rho) + (1-\rho)\log(1-\rho)},\end{aligned}$$

from which

$$\begin{aligned}\lim_{\rho\to 0}\frac{R(d)}{H(\rho)} &= 1 - \lim_{\rho\to 0}\frac{d\log(\rho d) - d\log(1-\rho d)}{\log(\rho) - \log(1-\rho)} \\ &= 1 - \lim_{\rho\to 0}\frac{d/\rho + d^2/(1-\rho d)}{1/\rho + 1/(1-\rho d)} \\ &= 1 - d.\end{aligned}$$

This shows that if we normalize the rate and the distortion by their maxima, $H(\rho)$ and ρ, respectively, the rate-distortion function becomes linear for sparse sources ($\rho \to 0$). Rate and distortion are fixed values for a certain instance of the set membership representation problem. The fact, that the rate-distortion becomes linear for small ρ and the fact, that the cropped filter is linear, give an intuitive reason for the optimality of the cropped filter for $\rho \to 0$. In reality, ρ is typically extremely small.

2.8 Hamming Distance and Hamming Distortion

The *Hamming distance* between two bit-vectors is the number of symbols that disagree. The *Hamming distortion* is the Hamming distance per symbol, thus a normalized Hamming distance.
The Hamming distance and at the same time the Hamming distortion of two elements x and $\hat{x}$ is given by

$$d(x,\hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x}, \\ 1 & \text{if } x \neq \hat{x}. \end{cases}$$

For this measure, the expected distortion is equivalent to the probability of a mistake. Consider a bitmap $x = x_1, ..., x_n$ of size n constructed from a set S, where each x_i is set to 1 if $x_i \in S$ and zero otherwise. Similarly associate a bitmap $\hat{x}$ with a set $\hat{S}$. The expected distortion between the two is then

$$d(S,\hat{S}) = \frac{1}{n}\sum_{i=1}^{n} d(x_i,\hat{x}_i).$$

So the distortion between two sets is the average of the per symbol distortion of the individual elements.

2.9 Comparison of Different Methods

When analyzing applications that use compact set membership representations, it is necessary to know the probability of a membership query being answered incorrectly, thus the probability of an error. We use the *Hamming distortion* as described above that precisely expresses this probability of error for the subsequent analysis and comparisons.

In order to express the distortion of compact set membership representations and to compare the existing methods to the theoretical distortion limit, a clearly defined model is needed. The model usually described when talking about Bloom filters allows for an infinite document space whereas in the model for cropped filters a bounded but arbitrarily big amount of possible document IDs is allowed. These differences make it challenging to compare the two.
To understand the basic case, as well as for simplicity of analysis, the first approach considers the case where the probability of an element being in a set is independently and identically distributed. This corresponds to the Bernoulli-ρ model as described in Sect. 2.3.3. Such a model may be appropriate, when identifiers are composed of a hash of the entire space.
Neglecting any possible correlation in the input will show Bloom filters in a better light performance-wise than they otherwise would. This is because Bloom filters inherently neglect any correlation by hashing it out.
Further, we assume a finite, enumerable space of possible elements containing the IDs $1..n$. m is the number of bits of the compact set representation.

2.9.1 Derivation of Distortion Formulas

We now derive the expected distortion D for several techniques of compactly representing the subset S as well as the absolute lower bound using the information rate-distortion function.

Bloom Filter

Since our model describes a bounded document space, we apply the Bloom filter on a bounded document-ID space. This approach is reasonable in many distributed systems since document IDs are often defined as some hash value of the whole document projecting them into a bounded space or are restricted to a value smaller than 2^{32} or 2^{128} by the application.
First, we look at the false positive rate of Bloom filters [BM02]. After inserting k documents into a Bloom filter of size m using h hash functions, the probability of one bit still set to zero is:

$$\Pr(\text{Bit still } 0) = \left(1 - \frac{1}{m}\right)^{hk}$$

The false positive rate is defined as the probability that for a document not inserted in the Bloom filter, all bits for all h hash functions are set to one. This equals to the probability of h random bits being set to one:

$$\begin{aligned}\Pr(\text{False positive}) &= (1 - \Pr(\text{Bit still } 0))^h \\ &= \left(1 - \left(1 - \frac{1}{m}\right)^{hk}\right)^h \\ &\approx \left(1 - e^{-hk/m}\right)^h\end{aligned}$$

We analyze the expected Hamming distance. In the Bernoulli model we do not know how many elements are actually contained in the Bloom filter. We therefore have to use conditional probabilities and sum up all possible occurrences for K being the random variable for the actual size of the subset.

$$\begin{aligned}
\mathbf{E}\left[d(x,\hat{x})\right] &= \mathbf{E}\left[\sum_{i=1}^{n} d(X_i,\hat{X}_i)\right] \\
&= \sum_{k=0}^{n} \Pr\left(K=k\right)\mathbf{E}\left[\sum_{i=1}^{n-k} d(X_i,\hat{X}_i)|K=k, X_i=0\right] \\
&= \sum_{k=0}^{n} \Pr\left(K=k\right)\mathbf{E}\left[\sum_{i=1}^{n-k} d(0,\hat{X}_i)|K=k, X_i=0\right] \\
&= \sum_{k=0}^{n} \Pr\left(K=k\right)\sum_{i=1}^{n-k}\mathbf{E}\left[d(0,\hat{X}_i)|K=k, X_i=0\right] \\
&= \sum_{k=0}^{n} \Pr\left(K=k\right)\sum_{i=1}^{n-k}\Pr\left(\hat{X}_i=1|K=k, X_i=0\right) \\
&= \sum_{k=0}^{n} \Pr\left(K=k\right)(n-k)\Pr\left(\text{False positive}|K=k\right) \\
&= \sum_{k=0}^{n} \binom{n}{k}\rho^k(1-\rho)^{(n-k)}(n-k)\Pr\left(\text{False positive}|K=k\right)
\end{aligned}$$

According to [BM02], the expected error rate is minimized for $h=\frac{m}{k}\ln 2$ hash functions. In practice, an integer number must be chosen, usually $h=\lfloor\frac{m}{k}\ln 2\rfloor$, as this requires less computation.
Therefore it follows that the normalized distortion is

$$\begin{aligned}
D_{\text{Bloom}} &= \sum_{k=0}^{n} \Pr\left(K=k\right)\frac{n-k}{n}\Pr\left(\text{False positive}|K=k\right) \\
&= \sum_{k=0}^{n} \binom{n}{k}\rho^k(1-\rho)^{(n-k)}\frac{n-k}{n}\left(1-e^{-hk/m}\right)^h.
\end{aligned}$$

Compressed Bloom Filters

Using compressed Bloom filters, we can in theory achieve a false positive rate arbitrarily close to $(0.5)^{\frac{m}{k}}$ by letting the number of hash functions go to 0 or infinity [Mit01].
In analogy to the derivation of the distortion for Bloom filters, we find the distortion for compressed Bloom filters to be:

$$\begin{aligned}
D_{\text{compressed Bloom}} &= \sum_{k=0}^{n} \Pr\left(K=k\right)\frac{n-k}{n}\Pr\left(\text{False positive}|K=k\right) \\
&= \sum_{k=0}^{n} \binom{n}{k}\rho^k(1-\rho)^{(n-k)}\frac{n-k}{n}\left(\frac{1}{2}\right)^{\frac{m}{k}}
\end{aligned} \tag{2.5}$$

Cropped Filter

For a cropped filter, we have the following expected Hamming distance for $\rho \leq 0.5$:

$$\begin{aligned}
\mathbf{E}\left[d(\hat{x}, x)\right] &= \mathbf{E}\left[\sum_{i=1}^{n} d(X_i, \hat{X}_i)\right] \\
&= \mathbf{E}\left[\sum_{i=1}^{n-m} d(X_i, 0)\right] \\
&= \sum_{i=1}^{n-m} \mathbf{E}\left[d(X_i, 0)\right] \\
&= \sum_{i=1}^{n-m} \Pr\left(X_i = 1\right) = (n-m)\rho
\end{aligned}$$

For the expected Hamming distance as above and a total number n of decoded bits we have a distortion of:

$$D_{\text{cropped}} = \frac{n-m}{n}\rho$$

Compressed Cropped Filters

Assuming an optimal compression algorithm, a cropped filter of size q can be compressed into a compressed cropped filter of size m as described by

$$m = qH(\text{Bit of the cropped filter}) = qH(\rho).$$

Therefore, we can describe q as

$$q = \frac{m}{H(\rho)}.$$

To find the distortion, we first look at the expected Hamming distance

$$\begin{aligned}
\mathbf{E}\left[d(x, \hat{x})\right] &= \mathbf{E}\left[\sum_{i=1}^{n} d(X_i, \hat{X}_i)\right] \\
&= \mathbf{E}\left[\sum_{i=1}^{n-q} d(X_i, 0)\right] \\
&= \sum_{i=1}^{n-q} \mathbf{E}\left[d(X_i, 0)\right] \\
&= \sum_{i=1}^{n-q} \Pr\left(X_i = 1\right) = (n-q)\rho.
\end{aligned}$$

Therefore, the distortion is

$$\begin{aligned}
D_{\text{compressed cropped}} &= \frac{n-q}{n}\rho \\
&= \frac{n - \frac{m}{H(\rho)}}{n}\rho \\
&= \frac{nH(\rho) - m}{nH(\rho)}\rho = \left(1 - \frac{m}{nH(\rho)}\right)\rho.
\end{aligned}$$

Cropped Filter With No False Negatives

In order to find the cost of an additional method having a one-sided error, we use the trivial method of the cropped filter to construct a filter that has no false negatives. One must decode all unknowns to one in order to omit any false negatives. We analyze the expected Hamming distance:

$$\begin{aligned}
\mathbf{E}\left[d(x,\hat{x})\right] &= \mathbf{E}\left[\sum_{i=1}^{n} d(X_i,\hat{X}_i)\right] \\
&= \mathbf{E}\left[\sum_{i=1}^{n-m} d(X_i,1)\right] \\
&= \sum_{i=1}^{n-m} \mathbf{E}\left[d(X_i,1)\right] \\
&= \sum_{i=1}^{n-m} \Pr\left(X_i=0\right) = (n-m)(1-\rho)
\end{aligned}$$

For the expected Hamming distance as above and a total number n of decoded bits we have a distortion of

$$D_{\text{cropped no false negatives}} = \frac{n-m}{n}(1-\rho).$$

Compressed Cropped Filter With No False Negatives

As with the cropped filter, we can also apply an additional compression step here. As for the above description for compressed cropped filters the equation

$$q = \frac{m}{H(\rho)}$$

holds. To find the distortion, we first look at the expected Hamming distance:

$$\begin{aligned}
\mathbf{E}\left[d(x,\hat{x})\right] &= \mathbf{E}\left[\sum_{i=1}^{n} d(X_i,\hat{X}_i)\right] \\
&= \mathbf{E}\left[\sum_{i=1}^{n-q} d(X_i,1)\right] \\
&= \sum_{i=1}^{n-q} \mathbf{E}\left[d(X_i,1)\right] \\
&= \sum_{i=1}^{n-q} \Pr\left(X_i=0\right) = (n-q)(\rho-1)
\end{aligned}$$

Therefore, the distortion is

$$\begin{aligned}
D_{\text{compressed cropped no false negatives}} &= \frac{n-q}{n}(\rho-1) \\
&= \frac{n-\frac{m}{H(\rho)}}{n}(\rho-1) \\
&= \left(1-\frac{m}{nH(\rho)}\right)(\rho-1).
\end{aligned}$$

We note that the difference between the 'compressed cropped filter' and the 'compressed cropped filter with no false negatives' is a constant factor of

$$\triangle = \frac{\rho - 1}{\rho}.$$

Lower Bound

We find an absolute lower bound that defines how big the rate of a communication must be in order to be able to transmit a given amount of information at a given fidelity [Ber71]. The function defining this bound is called the information rate-distortion function and can also be used to determine how much information about a source can be transmitted using a given rate. The rate R is the number of bits in the encoded sequence per bit from the input. In other words, it is the compression ratio. When the distortion is zero, the source may be encoded losslessly. According to [CT91], the information rate-distortion function for a Bernoulli-ρ source with Hamming distortion is given by

$$R(D) = \begin{cases} H(\rho) - H(D), & 0 \leq D \leq \min\{\rho, 1-\rho\}, \\ 0, & D > \min\{\rho, 1-\rho\} \end{cases}$$

and therefore the lowest achievable distortion is given by

$$D_{\text{lower bound}} = \begin{cases} H^{-1}\left(H(\rho) - R(D)\right), & 0 \leq R \leq H(\rho), \\ 0, & R > H(\rho) \end{cases}$$

where $H(\rho)$ is the entropy of a Bernoulli-ρ source, namely that $H(\rho) = -\rho \log \rho - (1-\rho)\log(1-\rho)$.

2.9.2 Visualization of the Distortion

The distortions of the methods of compact set membership representation deduced above as well as the lower bound are plotted in Fig. 2.4 for one fixed scenario. Several points spring to mind when looking at this graph:

- At around 352 bits, the curve denoting the lower bound hits the x-axis. This indicates, that for bigger sizes of the set membership data structure, lossless encoding is possible. All the same, some methods produce unnecessarily high error rates in this region.

- There clearly is a gap between the lower bound and the performance of a Bloom filter.

- The cropped filter with no false negatives performs rather poorly compared to the Bloom filter.

- The compressed cropped filter has a linear curve and touches the information rate-distortion function in the points for $m = 0$ and $D = 0$.

- As expected after studying the cost for having an only one-sided error, the methods that do not have a restriction on the error perform significantly better with respect to the distortion.

- The compressed cropped filter has the lowest expected error probability.

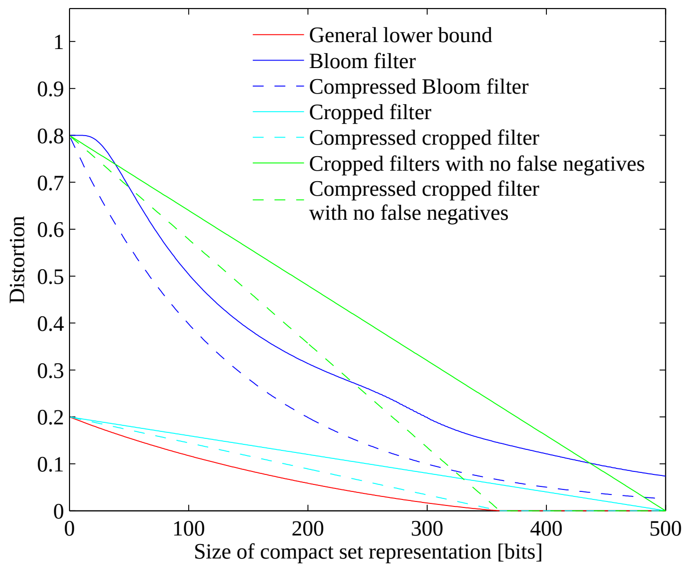

Figure 2.4: *Distortion for different methods of lossy set membership representation for a document space of* $n = 500$ *and a set membership probability of* $\rho = 0.2$.

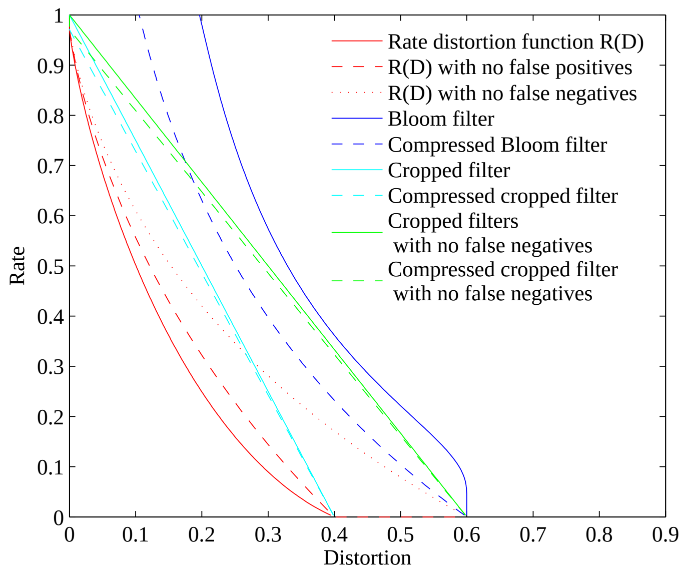

Figure 2.5: *Rate as a function of the distortion for different methods of lossy set membership representation for a document space of* $n = 300$ *and a set membership probability of* $\rho = 0.4$.

In information theory, rates are usually expressed and plotted as a function of the distortion of an encoding. According to [Say96], the *rate* of an encoding is defined as the average number of bits per symbol, thus $R = \frac{K}{N}$ for an encoding of an original sequence of length N into a compressed sequence of length K. Using this definition, we find the rate as

$$R = \frac{m}{n}.$$

The rates of all discussed methods over the distortion is plotted in Fig. 2.5. This graph is basically the same as in Fig. 2.4 but with normalized and swapped axes and therefore allows the same insights.

2.10 Cropped Bloom Filter

As a result of the feedback from the mid-thesis-presentation, a possible adaptation of Bloom filters to the case where false positives and false negatives are allowed was analyzed. As illustrated in Fig. 2.6, the document space is divided into parts A and B. The elements belonging to the subset in part A are inserted into a Bloom or compressed Bloom filter whereas the elements in B are ignored and thus become false negatives. In order to avoid the optimization of finding the best size of A and B for given parameters, we choose numerous different sizes for A and B and for each size of the final filter take the minimal distortion of all.
The distortion for a partition such that $|A| = r$ and $|B| = n - r$ is derived as follows. K represents the number of elements belonging to the subset in part A of the document space.

$$\begin{aligned}
\mathbf{E}\left[d(x, \hat{x})\right] &= \mathbf{E}\left[d(x_A, \hat{x}_A)\right] + \mathbf{E}\left[d(x_B, \hat{x}_B)\right] \\
\mathbf{E}\left[d(x_A, \hat{x}_A)\right] &= \sum_{k=0}^{r} \Pr(K = k) \mathbf{E}\left[d(x_A, \hat{x}_A) | K = k\right] \\
&= \sum_{k=0}^{r} \Pr(K = k)(r - k) \Pr(\text{False positive} | K = k) \\
&= \sum_{k=0}^{r} \binom{r}{k} \rho^k (1 - \rho)^{(r-k)} (r - k) \Pr(\text{False positive} | K = k) \\
\mathbf{E}\left[d(x_B, \hat{x}_B)\right] &= |B|\rho = (n - r)\rho
\end{aligned}$$

Then, the distortion is:

$$D = \frac{\sum_{k=0}^{r} \binom{r}{k} \rho^k (1 - \rho)^{(r-k)} (r - k) \Pr(\text{False positive} | (K = k)) + (n - r)\rho}{n}$$

The reason for the distortion of the Bloom and the compressed Bloom filter being so large for small filter sizes is the vast amount of false positives generated. The cropped Bloom filter does not exhibit this characteristic since its distortion is bounded by ρ. A cropped Bloom filter could always code no information which would results in a distortion $D = \rho$ using a maximum likelihood decoder.
Figure 2.7 shows the distortion behavior of the cropped Bloom filter using a regular Bloom filter and one using a compressed Bloom filter to store membership information about the elements in part A. The dotted bright lines show the distortion each for a particular partition of the document space. The solid black line is the minimum of all, thus representing a cropped Bloom filter using regular Bloom filters. The dashed black line uses compressed Bloom filters instead of regular ones and therefore yields a slightly smaller distortion. We note the following:

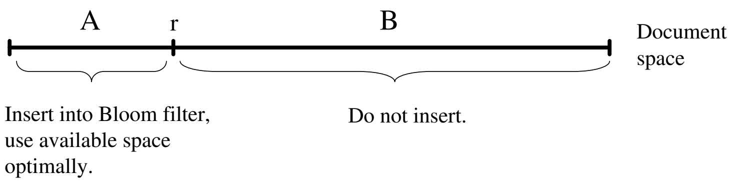

Figure 2.6: *Partition of the document space for the cropped Bloom filter.*

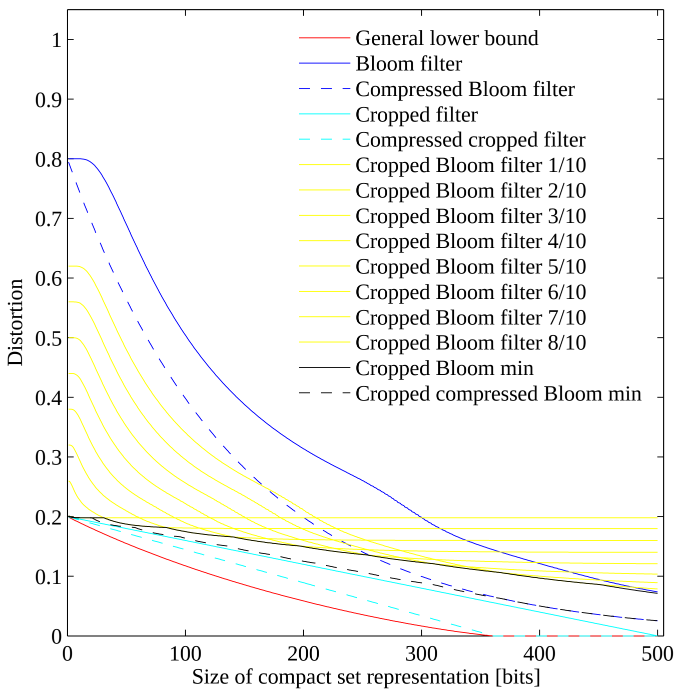

Figure 2.7: *The cropped Bloom filter is an adaptation of the classical Bloom filter to prevent the extreme growth of the distortion for small sizes of the data structure. The distortion of a cropped Bloom filter is plotted for a document space of size* $n = 500$ *and a document probability of* $\rho = 0.2$.

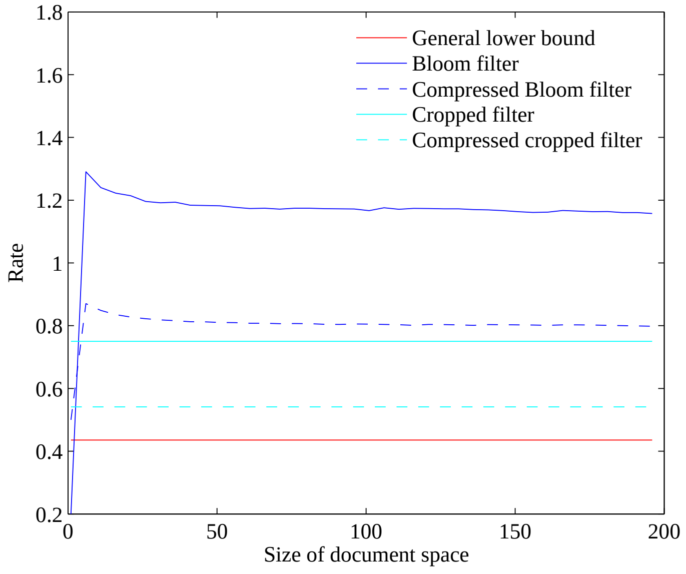

Figure 2.8: *Behavior of the rate for a changing size of the document space, a fixed distortion of $D = 0.5$, and a fixed document probability of $\rho = 0.2$. After some inconsistencies for very small document spaces, the rate is constant for all plotted methods of set membership representation.*

- The curve of the cropped Bloom respectively compressed cropped Bloom filter is tangent to the curve of the Bloom respectively the compressed Bloom filter.
- As the cropped Bloom filter is no longer restricted to having a one-sides error, its distortion for $m = 0$ equals to ρ.
- The cropped Bloom filter does not reach the minimal possible distortion as indicated by the general lower bound and it performs worse than the simple compressed cropped filter.

2.11 Rate vs. Document Space

An interesting consideration is how the rate behaves over a changing size of the document space for a given distortion. This is all the more necessary, since up to now, only small document spaces were taken into account. Figure 2.8 illustrates the rate behavior for the discussed methods of set membership representation and a fixed distortion of $D = 0.05$. We note that after some unevenness for very small document spaces, the rates are approximately constant.
The values for the Bloom and the compressed Bloom filter had to be calculated numerically. The other curves were derived from the equations in Sect. 2.9.1:

Cropped Filter

$$R = \frac{m}{n} = \frac{n - \frac{Dn}{\rho}}{n} = 1 - \frac{D}{\rho}$$

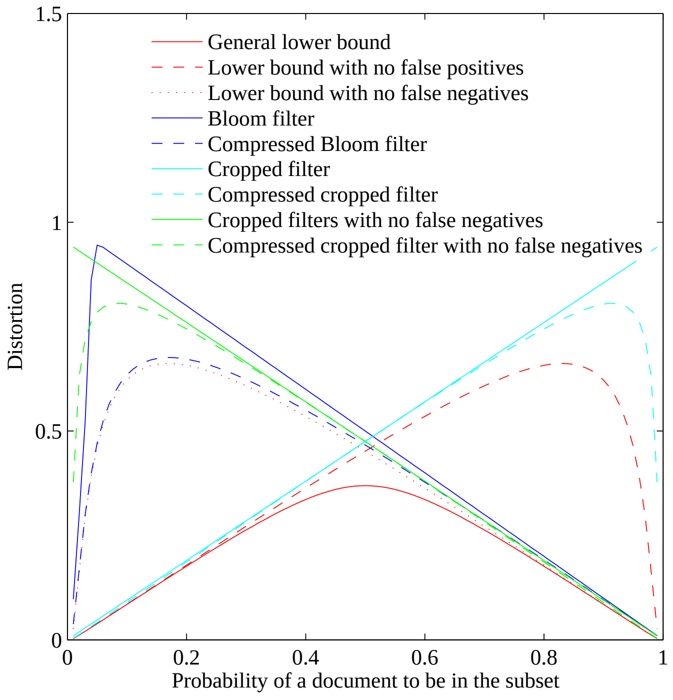

Figure 2.9: *Behavior of the distortion for a changing document probability, a fixed size of the data structure $m = 50$, and a fixed document space $n = 1000$. The symmetries of methods having only false positives vs. methods having only false negatives are nicely exhibited.*

Compressed Cropped Filter

$$R = \frac{m}{n} = \frac{nH(\rho)\left(1-\frac{D}{\rho}\right)}{n} = H(\rho)\left(1-\frac{D}{\rho}\right)$$

General Lower Bound

$$R(D) = \begin{cases} H(\rho) - H(D), & 0 \leq D \leq \min\{\rho, 1-\rho\}, \\ 0, & D > \min\{\rho, 1-\rho\} \end{cases}$$

As one can see, all these functions do not depend on the document space. The rates of the cropped filter, the compressed cropped filter, and the lower bound of the rate for a given distortion are therefore independent of the document space. Also noticeable is the fact that the Bloom filter's rate is greater than one. In order to reach a distortion of only 0.05 in this setting, the Bloom filter needs to have a rate greater than one in order to anticipate false positives.

2.12 Distortion vs. ρ

The previous section investigates the behavior of lossy set membership representations when the size of the document space changes. Here, we want to see how the distortion behaves for a changing document probability while the size of the set representation and the size of the document space remain fixed. Again, we use the equations in Sect. 2.9.1 and find formulas for the distortion depending on the probability of a document being in the set. For the lower bounds with one-sided error, numerical methods were needed.

Bloom Filter

$$D = \sum_{k=0}^{n} \binom{n}{k} \rho^k (1-\rho)^{n-k} \frac{n-k}{n} \left(1 - e^{\frac{-hk}{m}}\right)^h$$

for $h = \frac{m}{k} \ln 2$.

Compressed Bloom Filter

$$D = \sum_{k=0}^{n} \binom{n}{k} \rho^k (1-\rho)^{n-k} \frac{n-k}{n} \left(\frac{1}{2}\right)^{\frac{m}{k}}$$

Cropped Filter

$$D = \frac{n-m}{n} \rho$$

Compressed Cropped Filter

$$D = 1 - \frac{m}{nH(\rho)} \rho$$

Cropped Filter With No False Negatives

$$D = \frac{n-m}{n} (1-\rho)$$

Compressed Cropped Filter With No False Negatives

$$D = \left(1 - \frac{m}{nH(\rho)}\right)(1-\rho)$$

General Lower Bound

$$D = H^{-1}\left(H(\rho) - \frac{m}{n}\right)$$

In Fig 2.9 we observe much symmetry. The lower bound with no false positives mirrors the lower bound with no false negatives, the cropped filter with no false negatives mirrors the cropped filter with no false positives, and the lower bound itself is symmetric. The center of the symmetries is the vertical line where $\rho = 0.5$ in all cases.
As in previous plots, the compressed Bloom filter is close to the rate-distortion function with no false negatives and the compressed cropped filter for $\rho < 0.5$ respectively the compressed cropped filter with no false negatives for $\rho > 0.5$ is very close to the general lower bound. The later distance is biggest for $\rho = 0.5$ and

Area of Application	Approximate Size
ISBN	$10^{10} \approx 2^{33}$
EAN (European Article Number)	$10^{13} \approx 2^{43}$
Social security number	$10^{9} \approx 2^{30}$
Telephone number	$10^{10} \approx 2^{33}$
Member number of a club	$< 10^{4} \approx 2^{13}$
Bank account number	$10^{9} \approx 2^{30}$
Postal code	$10^{5} \approx 2^{16}$
Passport number	$10^{7} * 26 \approx 2^{28}$
Internal book reference number of a library	$< 10^{7} \approx 2^{23}$
IEEE membership number	$10^{9} \approx 2^{30}$
ACM membership number	$10^{7} \approx 2^{23}$
RFID	2^{64}
Numbering human beings	$7 * 10^{9} \approx 2^{33}$
On demand organizations and virtual teams	$< 10^{3} \approx 2^{10}$
URLs	1024 byte $\approx 2^{8000}$
Numbering documents available on the internet	$8 * 10^{9} \approx 2^{33}$

Table 2.1: *Application areas of compact set membership representations and the corresponding size of the document space.*

becomes smaller towards the extremes $\rho = 0$ and $\rho = 1$. For small values of ρ, the compressed cropped filter yields the smallest distortion.
Again, these results can be interpreted as an evidence that the Bloom filter is close to optimal if a one-sided error is required but that other methods such as the compressed cropped filter can be better for solving the general set membership representation problem.
If one combines the lower bound with no false positives and the lower bound with no false negatives to form a general one-sided lower bound, the distortion of the resulting curve has two maxima at around $\rho = 0.15$ and $\rho = 0.85$ and becomes smaller towards $\rho = 0$, $\rho = 0.5$, and $\rho = 1$. This shows that the price paid for a one-sided error heavily depends on ρ.

2.13 Distortion for Huge and Realistic Document Spaces

Until now we have analyzed various characteristics of different lossy set membership representations using relatively small document spaces (< 200). For realistic applications, the document space is around 2^{16} or 2^{128}. In Tab. 2.1 possible document identifiers in current realistic application areas of compact set representation as well as their sizes are listed.
The distortion behavior for Bloom and compressed Bloom filters contains a binomial factor that depends on the size of the document space which is not practical to calculate for values above 2^{20}. We use Stirling's approximation [Mac03, Bla03] to render this formula computable. The subsequent analysis is limited to compressed Bloom filters since Bloom filters can be seen as a special case of the compressed version. We start with (2.5)

$$D = \sum_{k=0}^{n} \binom{n}{k} \rho^k (1-\rho)^{(n-k)} \frac{n-k}{n} \left(\frac{1}{2}\right)^{\frac{m}{k}} .$$

The subsequent approximation is derived from Stirling's approximation of the factorial function [Mac03]

$$\begin{aligned} \binom{n}{k} &\equiv \frac{n!}{(n-k)!k!} \\ &\backsimeq 2^{nH(\frac{k}{n}) - \frac{1}{2}\log_2\left[2\pi n \frac{n-k}{n}\frac{k}{n}\right]} . \end{aligned}$$

Now we can approximate the distortion of the compressed Bloom filter

$$D \backsimeq \sum_{k=0}^{n} 2^{nH(\frac{k}{n}) - \frac{1}{2}\log_2\left[2\pi n \frac{n-k}{n}\frac{k}{n}\right]} \rho^k (1-\rho)^{n-k} \frac{n-k}{n} \left(\frac{1}{2}\right)^{\frac{m}{k}} .$$

Full of expectation the formula was typed into Matlab but we unfortunately were stranded once more. Matlab approximates values like 2^{1000} with infinity and $0 * 2^{1000} = 0 * inf = NaN$ is then regarded as not being a number!
An old but apparently not outdated approach worked: use of logarithms.

$$\begin{aligned} D &\backsimeq \sum_{k=0}^{n} 2^{nH(\frac{k}{n}) - \frac{1}{2}\log_2\left[2\pi n \frac{n-k}{n}\frac{k}{n}\right]} \rho^k (1-\rho)^{n-k} \frac{n-k}{n} \left(\frac{1}{2}\right)^{\frac{m}{k}} \\ &= \sum_{k=0}^{n} 2^{nH(\frac{k}{n}) - \frac{1}{2}\log_2\left[2\pi n \frac{n-k}{n}\frac{k}{n}\right] + k\log(\rho) + (n-k)\log(1-\rho) + \log(n-k) - \log(n) - \frac{m}{k}} \end{aligned}$$

It now is easily and quickly possible to plot graphs for ranges up to 2^{20}. Figure 2.10 shows the distortion behavior for a document space of $n = 2^{16}$. Since the curves resemble straight lines, the interesting part which is marked with a rectangle is shown enlarged in Fig. 2.11.
The vertical dotted black line marks the entropy of the problem in the Bernoulli model i.e. to the right of this line lossless compression is possible.
We see that the main characteristics identified while analyzing distortion behavior in small document spaces also hold for the big and realistic ones. With respect to distortion in the Bernoulli model, the compressed cropped filter is the most accurate one for a given size of the data structure. The Bloom filter exhibits a huge distortion for small sizes of the data structure that can be explained through the existence of many false positives in this area. But also for a size bigger than the entropy, the Bloom filter has some distortion even though lossless coding is theoretically possible.

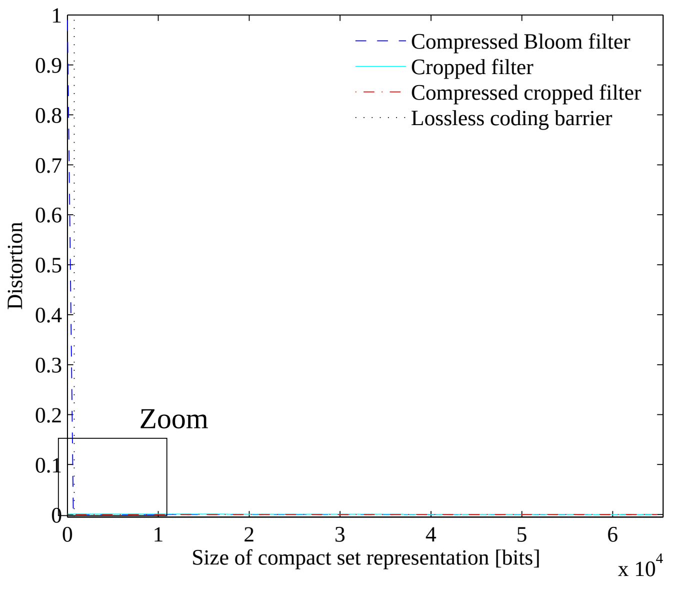

Figure 2.10: *Distortion for different methods of lossy set membership representation for a document space of size $n = 2^{16}$ and a set membership probability of $\rho = 0.001$. The zoomed-in graph of the interesting area can be found in Fig. 2.11.*

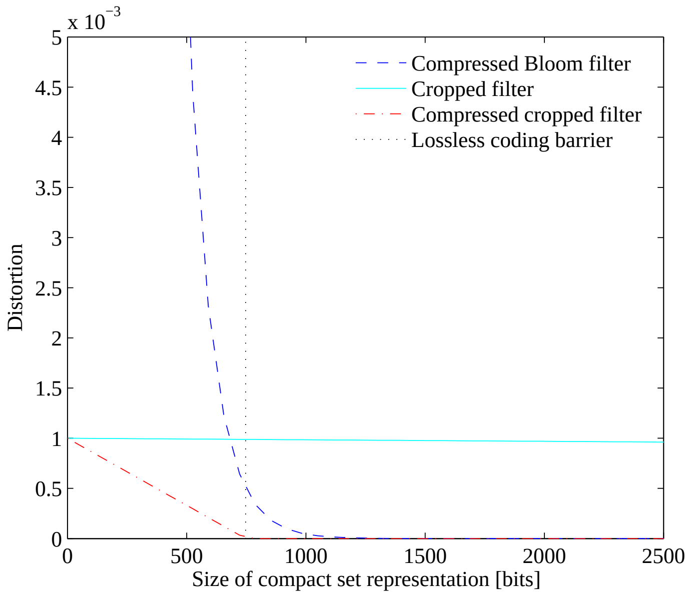

Figure 2.11: *Enlargement of interesting area of Fig. 2.10.*

Chapter 3

Simulation and Application Specific Issues

3.1 Simulation Setup

In order to test the behavior and applicability of the introduced methods of compact set representation, a trace-driven simulator of a distributed cache system was developed. One of the reasons for choosing a distributed caching environment was the fact that this problem does not rely on one-sided error. As previously mentioned, in a web environment, existing documents can disappear and new ones can appear at any time. Thus, the existence of false positives and false negatives is inherent to the scenario. Other reasons for choosing this scenario are its simplicity and the use of Bloom filters in existing distributed cache systems such as Summary Cache [FCAB00] which was then implemented in Squid v1.1.14 [Squ05].
The goal of the simulation is to compare the influence of different compact set membership representations on the response time in a distributed caching environment. The key component under study is the set representation a proxy keeps for its knowledge of the documents available in remote proxies that are part of a distributed cache. The performance of such a distributed cache using Bloom filters, compressed Bloom filters, one using cropped filters, and one using cropped filters combined with a frequency strategy is measured and compared.
The goal in a caching system is to minimize the response time. Since the focus of this evaluation is on the representation of documents, we only look at metrics of this subsystem that have an influence on the overall response time of the system. These are the correctness of the lookup results and the computation time per lookup.

3.1.1 Correctness of the Lookup Results

A false positive, namely a remote cache hit even though the remote cache does not contain the document, causes an additional superfluous request to the nearby cache. A false negative, namely a remote cache miss even though the remote cache contains the document, causes a request to the internet where a request to a nearby remote cache would have been sufficient. We define

- Cost of a request to the internet: *cInt*
- Cost of a request to a remote cache: *cLoc*.

And therefore

- Additional cost of a false positive: $cLoc$
- Additional cost of a false negative: $cInt - cLoc$.

3.1.2 Computation Time

The computation time has a direct influence on the total response time. Different methods of compact set membership representation cause variations in computation time for lookup operations. A prediction of the computation time would be:

$$\begin{aligned}&\text{Compressed copped filter}\\ &\quad > \text{Compressed Bloom filter}\\ &\qquad > \text{Bloom filter}\\ &\qquad\quad > \text{Cropped filter.}\end{aligned}$$

In networked applications, computation time is not the bottleneck, and we thus consider it negligible in comparison to network operations caused by false positives and negatives.

3.1.3 Workload

The workload plays a very important role in the whole system because it determines the theoretically possible hit and miss rate of the cache. It should meet several criteria [Jai91]:

Representativeness The workload should be close to real-life and there should be some correlation between the requests. Studies suggest it may obey a Zipf-like distribution.

Repeatability A workload should be such that the results can be reproduced easily.

Timeliness A workload should reflect the current usage patterns especially if this changes over the years.

The workload used was a trace of a real web-proxy. It is a sanitized log file from the CA*netII, Canada's coast to coast broadband research network, available at http://ardnoc41.canet2.net/cache/. This workload meets most of the above mentioned criteria. The data available was monitored in September 1999 and might therefore be slightly outdated and not perfectly reflect the patterns of cache requests of today and in the future, but all we were able to obtain.
We considered the information available at a node of the distributed cache to be uncorrelated to the others. If it was correlated, one could benefit from schemes as the Slepian-Wolf Coding [SW73].

3.2 Course of Action

One simulation phase consists of three stages illustrated in Fig. 3.1.

(1) Learning Phase In the beginning, 10 000 HTTP requests are sent to *Cache A* which stores the corresponding answers.

(2) Compress and Send *Cache A* creates a representation for all the identifiers of the stored resources and sends it to *Cache B*.

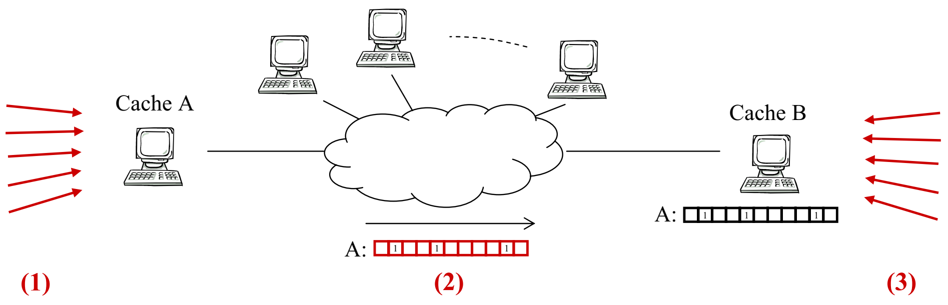

Figure 3.1: *Setup of the distributed cache simulation.*

(3) Query Phase 90 000 HTTP requests are sent to *Cache B*. Using the representation of the documents at *Cache A*, a check is made to see if a document is available from *Cache A*. The number of wrong answers is stored for different methods of set representation.

3.3 Implementing Compact Set Representation

The implementation of Bloom filters is straightforward. The URL of the HTTP request is taken as a string and the hashes calculated.
With cropped filters and compressed cropped filters, there are some implementation questions. One issue is that URLs can generally have a length of 1024 bytes and that storing this amount for each URL is not feasible. However, there are numerous possibilities for reducing the size of the label stored for each element:

Hash Function One approach is to use a hash function on the URLs to reduce the label size. This has the significant disadvantage of introducing hash collisions. The optimal size of such a hash value is determined with an optimization problem. In the optimum, a change in size of the hash range causes a change of collision probability that is equal to the marginal gain of an additionally stored document.

URL Compression This approach for reducing the label size assumes that URLs generally contain redundancy that can be removed using a compression scheme such as an LZ77/78-based coder or arithmetic coding.

Unlike Bloom filters, cropped filters have the choice of which elements to store. A clever strategy of which elements to pick can, depending on the nature of the queries, strongly improve the performance. If an incorrectly represented element is requested multiple times, one false conclusion causes several errors. Conversely, elements never requested by the application do not cause any error in this case.

Pick the First Few This is the cheapest method regarding computation cost. Another advantage is that the compact representation can be entirely constructed very quickly and is first available at other nodes.

Pick the Best Compressible Items This method causes the cropped filter to contain the most elements. When no information about the nature of the requests is available, this method produces the smallest expected error rate. However, it can be quite time consuming to find the best compressible items, it might even be np-complete.

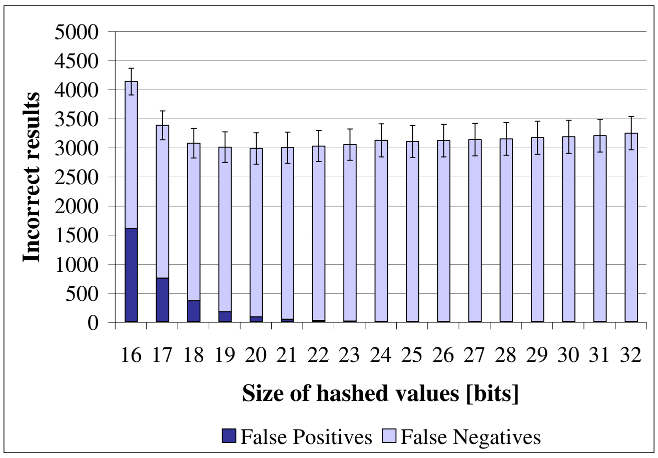

Figure 3.2: *Influence of the hash rage size on the correctness of the lookup results in a distributed cache.*

Pick the High Important Ones Using information or a prediction of the nature of the queries, the elements can be chosen cleverly in order to produce the lowest error. In a distributed cache environment, we would choose the elements that have the highest expected request frequency per storage bit in the compact representation. Other optimization metrics are conceivable such as download time, resource consumption, frequency of requests, temporal locality, or site locality.

3.4 Simulation Results

As mentioned in the previous section, there are several ways to implement a cropped filter. One possibility of reducing the label size, originally consisting of the entire URL, is to use a hash function. We examine the behavior of the resulting errors for different sizes of the hash range. Here, the set representation simply stores the outcome of the hash function for every request until a size of 40 000 bits is reached. The incorrect results for 90 000 subsequent requests can be found in Fig. 3.2. The trade-of collisions vs. size of a label is evident. We have an optimal collision to space consumption rate of around 20 bits for this specific setup.
The approach of compressing the original URLs using an existing compression scheme produces unfavorable results although the achievable compression rate is around 87%. Table 3.1 reveals the compression possibilities using WinZip. Hashing into a space of 32 bits yields a much better compression rate of around 99.3% and is practical for the given setting, since there are almost no collisions at all.
Unfortunately, the compressed cropped filters could not be implemented as described in Sect. 2.4. The close to optimal compression of a vector consisting of more than 2^{20} bits (depending on its construction) is not easily achievable. Therefore, we directly store the indexes of the bits set to 1 or, in other words, the document identifiers. Compressing them using existing compressors yields little gain only and is therefore abandoned.

Number of URLs	Size of compressed file	Compression rate
50 URLs	1.28 Kbyte	53%
100 URLs	2.24 Kbyte	60%
200 URLs	4.06 Kbyte	62%
300 URLs	5.62 Kbyte	65%
1 000 URLs	8.13 Kbyte	87%
10 000 URLs	88.32 Kbyte	85%

Table 3.1: *Benefit of compressing URLs using WinZip© 9.0 SR-1.*

In the subsequent simulation, the following methods of compact set representation are compared:

- **Cropped Filter** Stores the hash values of the requests using a hash range of 20 or 32 bits, whatever yields fewer errors depending on the size of the compact set representation. The elements are stored in their request order until the filter size is reached.
- **Frequency Cropped Filter** It is the same as a cropped filter but stores the elements that are most frequently requested first.
- **Bloom Filter** Deployed in its original version as described in [Blo70].
- **Compressed Bloom Filter** To our knowledge, compressed Bloom filters have not been implemented yet. We use their lower bound of false positives to find one for the total number of expected errors.

The size of the compact set membership representations is limited to 40 000 bits, the learning phase consists of 10 000, and the enquiry phase of 90 000 requests. The simulation was executed 20 times using a different trace in each run. The results can be found in Fig. 3.3.
For filter sizes up to 40 Kbits, all different implementations of the cropped filter outperform the Bloom filter as well as the compressed Bloom filter which exhibit a huge distortion. For filter sizes from 60 to 300 Kbits, the (compressed) Bloom filter produces less errors than a cropped filter and above 300 Kbits, both methods become error-free. There seems to be a critical size below which the amount of false positives of a Bloom filter becomes really large. This is not astonishing since it uses at least one hash function and after inserting around 10 000 elements into an array of comparable size, we expect most bits to be set to one which in turn causes lots of false positives.
Comparing the different versions of the cropped filter, one can clearly see that choosing the elements to be inserted into a cropped filter with a frequency strategy significantly reduces the resulting mistakes compared to just inserting the first requested elements. The difference is biggest for filter sizes around 40 Kbits.
The Bloom filter does not take advantage of specific knowledge about the nature of the requests. However, it automatically exploits the correlation between the inputs and the requests. Consider the possible false positives that could arise from an URL insertion into the Bloom filter. It is extremely likely, that these strings are not proper URLs and would thus never be requested. This results in lower false positives than if all inputs were equiprobable.
Depending on the application, it is sometimes desirable to represent some elements with a low error probability and in doing so decreasing the average error rate. This is a major advantage of the cropped filter.

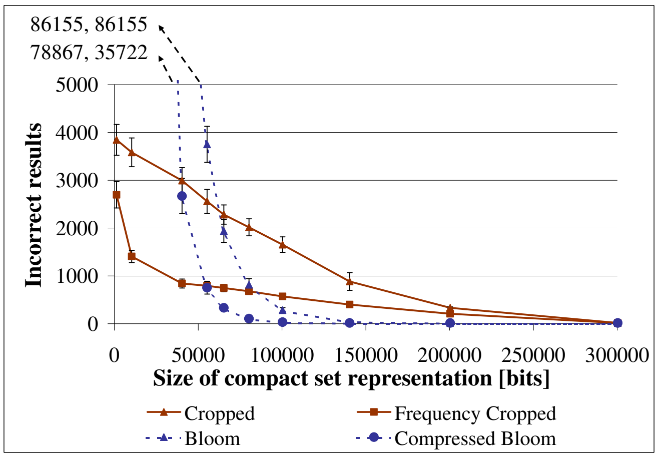

Figure 3.3: *Fidelity of lookup results in a distributed cache using different methods of compact set representation. The error bars reflect the 95% confidence intervals.*

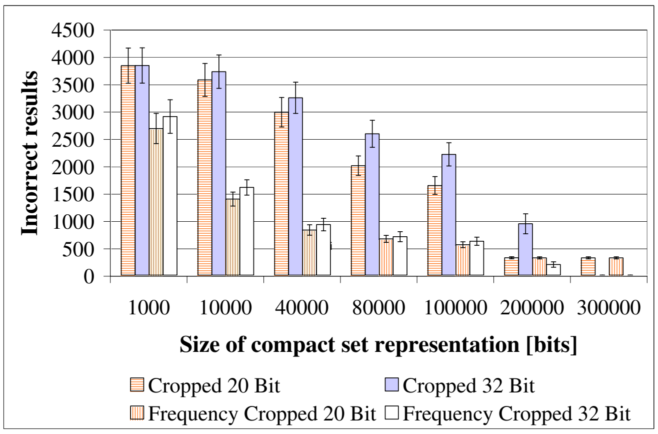

Figure 3.4: *The difference between the (frequency) cropped filter using a hash range of 20 bits and one using a hash range of 30 bits. The error bars reflect the 95% confidence intervals.*

Figure 3.4 illustrates the differences of a 20 bit hash range compared to a 30 bit hash range. The frequency cropped filter using 20 bits is better than the one using 32 bits up to a filter size of 100 Kbits. For a filter size of 200 Kbits, the effect of hash collisions becomes predominant and the 32 bits hash range is clearly favorable. This behavior can be explained with hash collisions that start to carry weight when the overall error becomes small for larger representation sizes. For smaller representation sizes, the fact that using a small hash range allows for more elements to be stored is predominant. When working without a frequency strategy, this effect is slightly shifted to the right and the 32 bits hash range is the better method for sizes above 300 Kbits.

Although the cropped filter does not perform as well as one would expect when looking at its theoretical performance in the Bernoulli model, it has a significantly lower error rate than a Bloom filter for some sizes of the compact set representation. Looking at the difference between the theoretical model in Chap. 2 and the simulation results, we conclude that the (compressed) cropped filter is especially good if the input set is uniformly distributed and the elements have a reasonably small label, when the document space is small. If a prior transformation is necessary, it should be minimized with respect to the loss of information e.g. collisions.

Using one hash function for reducing the document space is basically a special case of the Bloom filter, one with just one hash function. However, the Bloom filter typically uses an optimal amount of hash functions to prevent collisions and is therefore superior. It combines the two steps of reducing the document space and representing a set while optimizing the overall loss.

Cropped filters have a distinct advantage. Using a cropped filter, further messages containing information about more elements and thus resulting in more accuracy can easily be provided at any time whereas a Bloom filter needs to be fully constructed before transmission.

Chapter 4

Conclusions

Previously, compact set membership representation implied the use of Bloom filters. However, analysis shows that using Bloom filters means paying a large price for only false positives. The trivial method of storing some elements' identifiers and 'cropping' the information about the rest yields better error probabilities in the Bernoulli model and there might be other, more sophisticated methods. We therefore conclude that the use of Bloom filters is not optimal in every case and that it is worth considering alternatives before using them. This result may further be strengthened or weakened depending on the relative cost of false positives and false negatives.
Bloom filters remain strong when the application requires or benefits from restricting possible errors to false positives and when no a priori information is known about the type of the queries the filter will subsequently be used for. The cropped filter, on the other hand, is adaptable to incorporate such information. They can also be interpreted as erasure codes. Instead of simply guessing that all elements for which we have no information are not in the set, this information could be registered as missing - an erasure. Furthermore, their simplicity allows subsequent refinement of the fidelity by sending information of additional elements.
The master thesis at hand intends to open a new perspective in looking at compact set membership representation from an information theory viewpoint. I hope that such considerations propagate into the systems world and lead to more custom-made set membership data structures.

4.1 Future Work

Much future work remains. An open issue is the optimal and practical encoding of the cropped filter. A coding scheme being able to efficiently compress sparse and extremely big bit-vectors to a size close to the vector's entropy is needed. Our analysis uses equiprobable set membership as input model. Other input models with correlation as for example a Markov or hidden Markov model can be investigated. In our simulations, we considered the data available at each cache independent of each other, thus we did not yet exploit a possible correlation that may exists between the content of different caches. One could use schemes as the Slepian-Wolf Coding [SW73]. Also, a framework to build methods of compact set membership representation matching application specific tolerance for false positives and false negatives would be very useful. Furthermore, future work includes a more in-depth analysis of the effects of reducing the global set using hash functions. The optimal size of the hash range depends on application specific settings such as the number of elements in the subset and the cost of a collision.

4.2 Personal Experiences

From the beginning, my motivation for this work was extremely high. First of all, this was the first time for me as an ETH student that I was able to conduct open research and to push knowledge into a direction that did not exist before. This was especially challenging since neither me nor my supervisors knew how far and in what exact direction this work would lead during my six months thesis. Another reason for my motivation was the surrounding at the IBM research laboratory. After getting to know to ETH for more than four years and working part-time as a work-student for IBM Switzerland, it was exciting to experience an industrial research lab and to see both the differences to research at university and the differences to working in industry. Last but not least, the fact that the topic of optimizing compact set membership representation for distributed computing nicely combines my primary interests in distributed systems and in information theory, was reason for motivation.

During the first two months and every now and then during the rest of my thesis, I read books, papers, and reports. While working on the analysis and the cropped filter, it was surprising to me how irregular progress in research is. Sometimes I contributed more to the progress of my thesis during a bus ride home or before going to bed than in several days of sitting in front of the computer.

Having three supervisors at the same time was not always easy. It was challenging to comply with all expectations and it meant some additional administrative effort. On the other hand, I was able to profit from numerous advice and talking with more people about the same issues opened different perspectives and viewpoints.

Help and coaching is probably one of the main ingredients to a successful thesis. I am proud that there are two scientific publications emerging from this work and this is due to the guidance I've experienced. The past six months went by very quickly and I definitely learned a lot.

Appendix A

Mid Thesis Report

A.1 Achievements

I started my master thesis here in Rüschlikon on the 7th of September 2004. First, I was introduced to the Research Lab and set up my working environment. My task description included a wide range of possible focus areas and allowed for explorative research.
Throughout my work I was in close contact with my advisers Marcel Waldvogel, Paul Hurley, and Roger Wattenhofer with which regular meetings were held. The following subsections summarize the work done and goals achieved in the past months:

September

- Introductory reading of numerous papers.
- Started looking at the problem of finding the best linear method for lossy but compact set representation in order to optimize search in distributed network storage.
- Wrote some smaller simulations with Matlab.

October

- Theoretically solved the problem of linear compact set representation for a small test case. The following hypothesis resulted: In the linear case, the identity matrix filled up with zeros is as good as one can get.
- Further tested the above hypothesis with simulations in Matlab.
- Prepared and discussed further time plan as well as first outline of report.
- Read about information theory, distortion, and rate-distortion.
- Compared various existing as well as our 'new' method of compact set representation with respect to the expected bit error probability (= distortion).
- Agreed on giving a talk about "Compressed Bloom Filters" by Mitzenmacher and "The Bloomier Filter" by Chazelle, Kilian, Rubinfeld, and Tal in the IBM Research Lab on 5.11.2004.
- Thoroughly read the two papers and prepared slides.

November

- Mid-thesis-presentation at ETH.
- Gave talk on Bloom Filters and Other Compact Set Representations at IBM Research Lab.
- Found parametric expressions for plotting rate vs. distortion graphs of bloom filters, compressed bloom filters, cropped filters, and compressed cropped filters. Rate vs. distortion graphs are commonly used in information theory opposed to the previous graphs that plotted the size of compact set representation vs. the bit error probability. I also corrected some mistakes (use of conditional probability instead of expectation).
- Examined related work by Claudio Weidmann and Martin Vetterli. With its help I was able to plot the rate-distortion for the deterministic case when the exact number of documents in a subset is known.
- Continued writing the report.
- As result of the feedback from the mid-thesis-presentation I analyzed a possible adaptation of bloom filters to the case where false positives and false negatives are allowed.

December

- Added information rate-distortion curves for the cases where only false positives and only false negatives are allowed.
- Reviewed the paper "On Conspiracies and Hyperfairness in Distributed Computing" by Hagen Völzer for the ICDCS conference.

A.2 Future Work

As I have experienced in the first part of my thesis it is always difficult to predict the future and to foresee where things will lead. However, the following tasks are firmly scheduled:

- Finish the chapter on the theoretical part of improved coding that contains all work and insights gained in the previous three months.
- Analyze different methods of compact set membership representations in different scenarios.
- Derive rules when to best apply which method of compact set membership representation.
- Complete the report of my master thesis and prepare for the final presentations.

Finally, I would like to use this opportunity to wish Marcel Waldvogel all the best for his new position as a professor at the university of Konstanz.

Appendix B

Summary

Set membership representation is a data structure widely used in a great variety of applications. It stores information as to whether an element is contained in a set or not. Compact or lossy set membership representations can significantly reduce the memory usage, but introduce an error rate. The method of first and last resort used today is the Bloom filter dating back to 1970 and having the strong characteristic of only producing false positives.
The goal of this thesis is to optimize compact set membership representations. This is done by first carefully analyzing the existing method and later proposing the *compressed cropped filter* that outperforms the Bloom filter in at least some area of the parameter space.
In the initial analysis, the rate-distortion curve for the general set membership problem for equiprobable set membership is derived. It defines an absolute lower bound of the error rate. A comparison of this lower bound to the performance of the Bloom filter shows that there is lots of room for improvement. Further investigating the reason for the Bloom filters performance being poorer than expected, we quantify the cost of only having a one-sided error using a rate distortion function for the case when only false positives are allowed. And indeed, the loss of performance can for a large part be explained by this restriction to only false positives.
We introduce the *cropped filter*, a method of compact set representation without restriction on the nature of errors. The output of a source that emits a 0 for every element not contained in the set and a 1 for every element contained in the set is stored until the maximum size of the data structure is reached. Information about the remaining elements is not stored. The resulting lossy representation contains perfect information about the stored area of the document space and uses a maximum likelihood estimation for the remaining documents. The *compressed cropped* filter goes one stage further and compresses the output of the source using a close to optimal compressor such as arithmetic coding. We find that the resulting error rate is close to the absolute lower bound for compact set representation assuming equiprobable set membership.
To test the behavior and applicability of the introduced methods, we use a trace driven simulator of a distributed cache system. For small sizes of the data structure, the cropped filter produces fewer incorrect results than the Bloom filter. When choosing the elements that are stored in the cropped filter with a frequency strategy, the result is further improved. However, the cropped filter does not perform as well as one would expect when looking at the theoretical performance derived in the initial analysis. Possible explanations include the non equiprobable distribution of the URLs, a correlation between the stored and the requested elements, and the necessity of reducing the document space, in our simulation performed using a hash function.

This work opens a new perspective in looking at compact set representation from an information theory viewpoint and questions the prevailing religious attitude towards Bloom filters.

Appendix C

Zusammenfassung

Die Speicherung von Mengenzugehörigkeitsinformationen gelangt in vielen Applikationen zur Anwendung. Kompakte oder verlustbehaftete Speicherung von Mengenzugehörigkeit erlaubt eine drastische Verringerung des benötigten Speicherbedarfs und hat eine verhältnismässig geringe Fehlerrate zur Folge. Für diesen Zweck wird zurzeit fast ausschliesslich der so genannte *Bloomfilter* verwendet, der im Jahre 1970 erfunden wurde und welcher die möglichen Fehler auf falsch positive beschränkt.
Das Ziel dieser Masterarbeit ist die Verbesserung der Methoden zur Speicherung von Mengenzugehörigkeitsinformationen. Zuerst wird der bestehende Bloomfilter analysiert und anschliessend der Cropped-Filter eingeführt, welcher ersteren unter gewissen Bedingungen übertrifft.
In der anfänglichen Analyse wird die *Rate-Distortion-Kurve für die allgemeine Speicherung von Mengenzugehörigkeitsinformationen* bei gleichwahrscheinlicher Mengenzugehörigkeit hergeleitet. Diese Kurve stellt eine absolute untere Schranke der Fehlerrate dar. Wird diese untere Schranke mit dem Verhalten des Bloomfilters verglichen, ist eine grosse Differenz ersichtlich, welche auf Verbesserungs- möglichkeiten schliessen lässt. Auf der Suche nach Gründen für die oben erwähnte Differenz gelang es uns, die Kosten für die Beschränkung der möglichen Fehler auf nur falsch positive mit einer *Rate-Distortion-Kurve für Methoden mit nur falsch positiven Fehlern* zu quantifizieren. Und tatsächlich, ein Grossteil der Performanz-Einbusse des Bloomfilters ist dem einseitigen Fehlerverhalten zuzuschreiben.
Der *Cropped-Filter* wird als Methode eingeführt, welche das Fehlerverhalten nicht beschränkt. Der Output einer Quelle, welche für jedes in der Menge enthaltene Element eine Eins und für jedes nicht enthaltene Element eine Null ausgibt, wird direkt gespeichert, bis der zur Verfügung stehende Speicherplatz aufgebraucht ist. Informationen über die restlichen Elemente werden verworfen (engl.: to crop = abschneiden). Die entstehende verlustbehaftete Mengenrepräsentation enthält alle Informationen über den gespeicherten Bereich des Dokumentenraums und für Elemente im verbleibenden Raum wird eine Maximum-Likelihood-Schätzung verwendet. Der *komprimierte Cropped-Filter* geht einen Schritt weiter und komprimiert den Output der oben erwähnten Quelle mit einem möglichst perfekten Kompressor wie z.B. Arithmetischem Codieren. Die resultierende Fehlerwahrscheinlichkeit ist nahe an der Rate-Distortion-Kurve für die allgemeine Speicherung von Mengenzugehörigkeitsinformationen bei gleichwahrscheinlicher Mengenzugehörigkeit.
Um das Verhalten und die Anwendbarkeit der beschriebenen Methoden zu evaluieren, wurde ein Simulator eines verteilten Caches erstellt. Für Datenstrukturen der Mengenzugehörigkeit von kleiner Grösse schneidet der Cropped-Filter mit einer kleineren Fehlerrate ab als der Bloomfilter. Wenn man zusätzlich die Elemente, welche im Cropped-Filter fehlerfrei gespeichert werden, mit einer Frequenzstrategie auswählt, wird das Resultat abermals verbessert. Dennoch ist das Verhal-

ten des Cropped-Filters nicht so gut wie die anfängliche Analyse erwarten liess. Mögliche Gründe sind die nicht gleichwahrscheinliche Verteilung der URLs, eine Korrelation zwischen den gespeicherten und den abgefragten Elementen sowie die Notwendigkeit, die Grösse des Dokumentenraums zu verkleinern. Letzteres wurde in der Simulation mit einer Hash-Funktion gemacht.
Die vorliegende Masterarbeit soll neue Perspektiven eröffnen, indem sie Datenstrukturen verlustbehafteter Mengenzugehörigkeit aus dem informationstheoretischen Standpunkt beleuchtet und mit den gewonnen Erkenntnissen den vorbehaltlosen Einsatz von Bloomfiltern in Frage stellt.

Bibliography

[BCMR02] John Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. Informed content delivery across adaptive overlay networks. In *Proceedings of ACM SIGCOMM*, pages 47–60, Pittsburgh, Pennsivania, USA, October 2002.

[Ber71] Toby Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.

[BHPW04] Daniel Bauer, Paul Hurley, Roman Pletka, and Marcel Waldvogel. Bringing efficient advanced queries to distributed hash tables. In *Proceedings of IEEE LCN*, November 2004.

[Bla03] Richard E. Blahut. *Algebraic Codes for Data Transmission.* Cambridge University Press, Cambridge, UK, 2003.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[BM02] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. In *Proceedings of the 40th Annual Allerton Conference on Communications, Control, and Computing*, pages 636–646, 2002.

[CKRT04] Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: An efficient data structure for static support lookup tables. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 30–39. Society for Industrial and Applied Mathematics, 2004.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., Hoboken, NJ, USA, 1991.

[DHW05] Andreas Diener, Paul Hurley, and Marcel Waldvogel. Bloom filters in networking: One size fits all? Research Report RZ-3591, IBM, 2005.

[FCAB00] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[HDW05] Paul Hurley, Andreas Diener, and Marcel Waldvogel. Lossy set compression: Bounds and algorithms. Research Report RZ-3592, IBM, 2005.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc., Hoboken, NJ, USA, 1991.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, Cambridge, UK, 2003.

[Mit01] Michael Mitzenmacher. Compressed bloom filters. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, pages 144–150. ACM Press, 2001.

[RK02] Sean C. Rhea and John Kubiatowicz. Probabilistic location and routing. In *Proceedings of INFOCOM 2002*, 2002.

[Say96] Khalid Sayood. *Introduction to Data Compression.* Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1996.

[Squ05] Squid. Squid web proxy cache. `http://www.squid-cache.org/`, 2005.

[SW73] David Slepian and Jack K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, IT-19:471–480, July 1973.

[WV99] Claudio Weidmann and Martin Vetterli. Rate-distortion analysis of spike processes. In *Proceedings of DCC'99*, 1999.

[WV01] Claudio Weidmann and Martin Vetterli. Rate distortion behavior of sparse sources. Technical report, EPFL, October 2001.