The Sketching Complexity of Pattern Matching

Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120, USA.
{ziv,jayram,robi,ravi}@almaden.ibm.com

Abstract. We address the problems of pattern matching and approximate pattern matching in the sketching model. We show that it is impossible to compress the text into a small sketch and use only the sketch to decide whether a given pattern occurs in the text. We also prove a sketch size lower bound for approximate pattern matching, and show it is tight up to a logarithmic factor.

1 Introduction

Pattern matching is the problem of locating a given (smaller) pattern in a (larger) text. It is one of the most fundamental problems studied in computer science, having a wide range of uses in text processing, information retrieval, computational biology, compilers, and web search. These application areas typically deal with large amounts of data and therefore necessitate highly efficient algorithms in terms of time and space.

In order to save space, I/O, and bandwidth, large text files are frequently stored in compressed form. The naive method for locating patterns in compressed files is to first decompress the files, and then run one of the standard pattern matching algorithms on them. Amir and Benson [2] initiated the study of pattern matching in compressed files; their approach is to process the compressed text directly, without first decompressing it. Their algorithm, as well as all the subsequent work in this area [3, 21, 12, 24, 11, 22, 15], deal with *lossless* compression schemes, such as Huffman coding and the Lempel-Ziv algorithm. The main focus of these results is the speedup gained by processing the compressed text directly.

In this paper we investigate a closely related question: how succinctly can one compress a text file into a small "sketch", and yet allow locating patterns in the text using the sketch alone? In this context we consider not only lossless compression schemes but also *lossy* ones. In turn, we permit pattern matching algorithms that are randomized and can make errors with some small constant probability. Our main focus is not on the speed of the pattern matching algorithms but rather on the succinctness of the compression. Highly succinct compression schemes of this sort could be very appealing in domains where the text is a massive data set or when the text needs to be sent over a network.

A fundamental and well known model that addresses problems of this kind is the *sketching model* [8, 14], which is a powerful paradigm in the context of computations over massive data sets. Given a function, the idea is to produce a fingerprint (*sketch*) of the data that is succinct yet rich enough to let one compute or approximate the function on the data. The parameters that play a key role in the applications are the size of the sketch, the time needed to produce the sketch and the time required to compute the function given the sketch.

Results. Our first main result is an impossibility theorem showing that in the worst-case, no sketching algorithm can compress the text by more than a constant factor and yet allow exact pattern matching. Specifically, any sketching algorithm that compresses any text of length n into a sketch of size s and enables determining from the sketch alone whether an input pattern of length $m = \Omega(\log n)$ matches the text or not with a constant probability of error requires $s \geq \Omega(n - m)$. We further show that the bound is tight, up to constant factors.

The proof of this lower bound turns out to be more intricate than one might expect. One of the peculiarities of the problem is that it exhibits completely different behaviors for $m \leq (1 - o(1)) \log n$ and $m \geq \log n$. In the former case, a simple compression of the text into a sketch of size 2^m is possible. We prove a matching lower bound for this range of m as well. These results are described in Section 3.

Our second main result is a lower bound on the size of sketches for *approximate pattern matching*, which is a relaxed version of pattern matching: (i) if the pattern occurs in the text, the output should be "a match"; (ii) if every substring of the text is at Hamming distance at least k from the pattern, the output should be "no match". An arbitrary answer is allowed if neither of the two holds. We prove that any sketching algorithm for approximate pattern matching, requires sketch size $\Omega(n/m)$, where n is the length of the text, m is the length of the pattern, and the Hamming distance at question is $k = \varepsilon m$, for a fixed $0 < \varepsilon < 1$. We further show that this bound is tight, up to a logarithmic factor. These results are described in Section 4.

Interestingly, Batu *et al.* [6] showed a *sampling* procedure that solves (a restricted version of) approximate pattern matching using $\tilde{O}(n/m)$ non-adaptive samples from the text. In particular, their algorithm yields a sketching algorithm with sketch size $\tilde{O}(n/m)$. This procedure was the main building block in their sub-linear time algorithm for weakly approximating the edit distance. The fact that our sketching lower bound nearly matches their sampling upper bound suggests that it might be hard to improve their edit distance algorithm, even in the sketching model.

Techniques. A sketching algorithm naturally corresponds to the communication complexity of a one-way protocol. Alice holds the text and Bob holds the pattern. Alice needs to send a single message to Bob (the "sketch"), and Bob needs to use this message as well as his input to determine whether there is a match or not.[1]

[1] Usually, a sketching algorithm corresponds to the communication complexity of a simultaneous messages protocol, which is equivalent to summarizing each of the text

The most classical problem which is hard for one-way communication complexity is the indexing function: Alice is given a string $\mathrm{x} \in \{0,1\}^n$ and Bob is given an index $i \in \{1,\ldots,n\}$, and based on a single message from Alice, Bob has to output x_i. It is well known that in any protocol solving this problem, even a randomized one, Alice's message has to be of length $\Omega(n)$. Our lower bound for approximate pattern matching is proved by a reduction from the indexing function.

Our lower bound for exact pattern matching uses a reduction from a variant of the indexing function. In this variant, Alice gets a string $\mathrm{x} \in \{0,1\}^n$; Bob gets an index $i \in [n]$ and also the $m-1$ bits preceding x_i in x; the goal is to output x_i. Using tools from information theory we prove an $\Omega(n-m)$ lower bound for this problem in the one-way communication complexity model.

Related work. Pattern matching and approximate pattern matching have a rich history and extensive literature—see, for instance, the excellent resource page [20]. To the best of our knowledge, pattern matching, has not been considered in the sketching model. For approximate pattern matching, the only relevant result appears to be the above mentioned work of Batu *et al.* [6].

Sketching algorithms for various problems, such as estimation of similarity between documents [8, 7, 9], approximation of Hamming distance [19, 13] and edit distance [4] between strings, and computation of L_p distances between vectors [1, 14], have been proposed in the last few years. Sketching is also a useful tool for approximate nearest-neighbor schemes [19, 16], and it is related to low-dimensional embeddings and to locality-sensitive hash functions [16].

2 Preliminaries

2.1 Communication complexity

A *sketching algorithm* is best viewed as a *public-coin one-way communication complexity protocol.* Two players, Alice and Bob, would like to jointly compute a two-argument function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. Alice is given $x \in \mathcal{X}$ and Bob is given $y \in \mathcal{Y}$. Based on her input and based on randomness that is shared with Bob, Alice prepares a "sketch" $s_A(x)$ and sends it to Bob. Bob uses the sketch, his own input y, and the shared randomness to determine the value $f(x,y)$. For every input $(x,y) \in X \times Y$, the protocol is required to be correct with probability at least $1-\delta$, where $0 < \delta < 1$ is some small constant. Typically, the error probability δ can be reduced by repeating the procedure several times independently (in parallel).

The main measure of cost of a sketching algorithm is the length of the sketch $s_A(x)$ on the worst-case choice of shared randomness and of the input x. Another

and the pattern into a small sketch. However, in the context of pattern matching, it is reasonable to have a weaker requirement, namely, that only the text needs to be summarized.

important resource is the amount of randomness between Alice and Bob. Newman [23] shows that the amount of shared randomness can always be reduced to $O(\log \frac{n}{\delta'})$ at the cost of increasing the protocol's error probability by δ'. In one-way protocols, Alice can privately generate these $O(\log \frac{n}{\delta'})$ and send them to Bob along with the sketch $s_A(x)$.

Some of our lower bounds use a reduction from the standard *indexing problem*, which we denote by IND_t: Alice is given a string $\mathrm{x} \in \{0,1\}^t$, Bob is given $j \in [t]$, and the goal is to output x_j. This problem has a lower bound of $t(1 - H_2(\delta))$ in the one-way communication complexity model [18, 5].

2.2 Pattern matching and approximate pattern matching

For a Boolean string $\mathrm{x} \in \{0,1\}^n$ and integer $1 \leq j \leq n$, let x_i denote the jth bit in x. For integers $1 \leq i \leq j \leq n$, $[i,j]$ denotes the corresponding integer interval, $[n]$ the interval $[1,n] = \{1,\ldots,n\}$, and $\mathrm{x}[i,j]$ denotes the substring of x that starts at position i and ends at position j. We define the pattern matching and approximate pattern matching problems in the communication model.

Let $0 \leq m \leq n$. In the (n,m) *pattern matching* problem, denoted $\mathrm{PM}_{n,m}$, Alice gets a string $\mathrm{x} \in \{0,1\}^n$ and Bob gets a string $\mathrm{y} \in \{0,1\}^m$. The goal is to determine whether there exists an index $i \in [n-m+1]$ such that $\mathrm{x}[i, i+m-1] = \mathrm{y}$. For the purpose of lower bounds, we would consider the simple Boolean function defined above. However, some of the algorithms we present can additionally find the position of the match i, if it exists.

We denote the *Hamming distance* of two strings $\mathrm{x}, \mathrm{y} \in \{0,1\}^n$ by $\mathrm{HD}(\mathrm{x},\mathrm{y}) \stackrel{\text{def}}{=} |\{i \in [n] : \mathrm{x}_i \neq \mathrm{y}_i\}|$. A relaxed version of pattern matching is the (n,m,ε) *approximate pattern matching* problem, denoted $\mathrm{APM}_{n,m,\varepsilon}$, in which Bob would like to determine whether there exists an index $i \in [n-m+1]$ such that $\mathrm{x}[i, i+m-1] = \mathrm{y}$, or whether for all $i \in [n]$, $\mathrm{HD}(\mathrm{x}[i, i+m-1], \mathrm{y}) \geq \varepsilon m$, assuming that one of the two holds.

Notation. Throughout the paper we denote random variables in upper case. For a Boolean string $\mathrm{x} \in \{0,1\}^n$, $|\mathrm{x}|$ denotes the Hamming weight (i.e., the number of 1's) of x. log denotes a logarithm to the base 2; ln denotes the natural logarithm. $H_2(p) = -p \log p - (1-p)\log(1-p)$ is the binary entropy function.

3 Exact pattern matching

In this section we obtain a simple sketching algorithm for exact pattern matching and show almost matching lower bounds. Recall that we denote by $\mathrm{PM}_{n,m}$ the problem in which Alice gets a string $\mathrm{x} \in \{0,1\}^n$, Bob gets a string $\mathrm{y} \in \{0,1\}^m$, and their goal is to find whether there exists an index $i \in [n-m+1]$ such that $\mathrm{x}[i, i+m-1] = \mathrm{y}$.

3.1 Upper bounds

First, we show an efficient (randomized) sketching algorithm for the pattern matching problem, based on the Karp–Rabin hash function [17]. Next, we show a deterministic sketching algorithm for the Boolean version of the pattern matching problem.

Proposition 1. *For $m \le n - \log n$, there is a one-sided error randomized sketching algorithm for the pattern matching problem $\mathrm{PM}_{n,m}$ using a sketch of size $O(n-m)$.*

Proof. The randomized algorithm is based on the Karp–Rabin method [17]. Let $t = n-m+1$; we assume in the sequel that $t \le n/3$, as otherwise the proof follows trivially by Alice sending x. Let $\mathrm{x}^1, \ldots, \mathrm{x}^t$ denote the sequence of t substrings of x of length m. Alice and Bob use the shared randomness to pick a (weak) 2-universal hash function $h : \{0,1\}^m \to [n^2]$. Alice sends to Bob $h(\mathrm{x}^1), \ldots, h(\mathrm{x}^t)$. Bob outputs "match found at i", if $h(\mathrm{x}^i) = h(\mathrm{y})$. If no such i exists, Bob outputs "no match found".

This is a one-sided error algorithm: if there is a match, it will surely be output. There is a possibility for a false match, though: when $\mathrm{x}^i \ne \mathrm{y}$, but $h(\mathrm{x}^i) = h(\mathrm{y})$. The probability for a false match is thus at most the probability h has a collision between y and any of $\{\mathrm{x}^1, \ldots, \mathrm{x}^t\}$. A union bound shows that since the range of h is large enough, the probability of a collision between y and any x^i is at most $O(1/n)$.

The scheme described above uses a sketch of size $O(t \log n) = O((n-m)\log n)$. Further improvement is possible using the Karp–Rabin hash function $h(b) = (\sum_{i=1}^{m} b_i \cdot 2^{m-i}) \bmod p$, where p is a randomly chosen prime in the range $[n^3]$ (here b_i is the ith bit in the binary representation of b). The advantage of this hash function is that the value of $h(\mathrm{x}^{i+1})$ can be computed from the value of $h(\mathrm{x}^i)$ and from the two bits x_i and x_{i+m}: $h(\mathrm{x}^{i+1}) = ((h(\mathrm{x}^i) - \mathrm{x}_i \cdot 2^{m-1}) \cdot 2 + \mathrm{x}_{i+m}) \bmod p$. Thus, what Alice needs to send is only $h(\mathrm{x}^1)$, the first t bits of x, and the last t bits of x. Thus, the sketch size goes down to $2t + O(\log n) = O(n-m)$.

Proposition 2. *There is a deterministic sketching algorithm for the pattern matching problem $\mathrm{PM}_{n,m}$ using a sketch of size 2^m.*

Proof. In the deterministic algorithm Alice sends to Bob a characteristic vector of length 2^m specifying all the strings of length m that occur as substrings of x. Bob outputs "match found" if and only if y is one of the substrings indicated by the characteristic vector.

3.2 Lower bounds

We show lower bounds on the sketch size for the pattern matching problem. The first one, Theorem 1, deals with the case $m \ge \Omega(\log n)$. The second one, Theorem 2, considers the case $m \le O(\log n)$.

Theorem 1. *If $n \leq m + \delta 2^m$, then any δ-error randomized sketching algorithm for the pattern matching problem* $\mathrm{PM}_{n,m}$ *requires a sketch of size at least* $(n - m + 1) \cdot (1 - H_2(2\delta))$*, where* $H_2(\cdot)$ *is the binary entropy function.*

Proof. Using Yao's Lemma [25], it suffices to exhibit a distribution μ over instances of $\mathrm{PM}_{n,m}$, and prove that any deterministic sketching algorithm that computes $\mathrm{PM}_{n,m}$ correctly with probability at least $1 - \delta$ when running over inputs chosen according to μ requires a sketch of size at least $(n - m + 1) \cdot (1 - H_2(2\delta))$.

The distribution μ is defined as follows. Alice is given a uniformly chosen bitstring $\mathrm{X} \in \{0,1\}^n$. Bob is given a uniformly chosen substring of X of length $m - 1$ concatenated with the bit 1.

The distributional lower bound w.r.t. μ is proven via a reduction from the following version of the indexing function, which we denote by $\mathrm{IND}_{n,k}$: Alice is given a string $\mathrm{x} \in \{0,1\}^n$, and Bob is given an index $k + 1 \leq j \leq n$ and a string $\mathrm{y} \in \{0,1\}^k$, which is promised to be equal to the substring $\mathrm{x}[j-k, j-1]$. The goal is to compute x_j.

Let ν be the following distribution over instances of $\mathrm{IND}_{n,k}$. Alice gets a uniformly chosen bitstring X in $\{0,1\}^n$; Bob gets an index J, which is chosen independently and uniformly in the interval $[k+1, n]$, and also the bits $\mathrm{X}_{J-k}, \ldots, \mathrm{X}_{J-1}$.

The following lemma shows the reduction.

Lemma 1. *Any deterministic sketching algorithm Π that computes* $\mathrm{PM}_{n,m}$ *with error probability at most δ on instances drawn according to μ yields a deterministic sketching algorithm Π' that computes* $\mathrm{IND}_{n,m-1}$ *with error probability at most 2δ on instances drawn according to ν and using exactly the same sketch size.*

Proof. In the indexing algorithm Π', given an input $\mathrm{x} \in \{0,1\}^n$, Alice sends whatever message she would have sent on this input in the algorithm Π. Given his input (j, y), where $m \leq j \leq n$ and $\mathrm{y} \in \{0,1\}^{m-1}$, Bob simulates the role of Bob in the pattern matching algorithm Π on the input $\mathrm{y} \circ 1$, where $\circ$ denotes the concatenation of strings. If the output in Π is "match found", Bob outputs "1" and otherwise he outputs "0".

It is easy to verify that when the input given to Π' is distributed according to ν, then the input given to Π in the reduction is distributed according to μ. We can thus assume that Π errs with probability at most δ.

Fix an input $(\mathrm{x}, (j, \mathrm{y}))$ for $\mathrm{IND}_{n,m-1}$, for which the protocol Π is correct on $(\mathrm{x}, \mathrm{y} \circ 1)$. If $\mathrm{IND}_{n,m-1}(\mathrm{x}, (j, \mathrm{y})) = 1$, then the string $\mathrm{y} \circ 1$ is a substring of x (at position $j - m + 1$), and thus Π' will output "1", as needed. Suppose then that $\mathrm{IND}_{n,m-1}(\mathrm{x}, (j, \mathrm{y})) = 0$. Clearly, $\mathrm{y} \circ 1$ does not equal to the substring of x that starts at position $j - m + 1$. Therefore, Π' outputs "0", unless there is some other substring of x that happens to equal to $\mathrm{y} \circ 1$. We next prove that the latter occurs with low probability.

Define E to be the set of instances $(\mathrm{x}, (j, \mathrm{y}))$, for which there exists some $i \neq j$, so that the substring $\mathrm{x}[i, i + m - 1]$ equals $\mathrm{y} \circ 1$. The proof of Lemma 1 would follow easily once we prove that $\Pr(E) \le \delta$, since Π' errs on an input $(\mathrm{x}, (i, \mathrm{y}))$ only if either it belongs to E or if Π errs on it.

Proposition 3. $\Pr(E) \le \delta$.

Proof. $\Pr(E)$ can be rewritten as $\Pr(\exists i \ne J \ : \ \mathrm{X}[i, i+m-1] = \mathrm{X}[J-m+1, J-1] \circ 1)$. In order to bound $\Pr(E)$, it would suffice to show that for all choices of $m \le j \le n$, $\Pr(\exists i \ne j \ : \ \mathrm{X}[i, i+m-1] = \mathrm{X}[j-m+1, j-1 \ \circ 1]) \le \delta$. So for the rest of the argument, fix such a j.

Define $t \stackrel{\text{def}}{=} j - m + 1$. We will show that for all $i \ne t$, $\Pr(\mathrm{X}[i, i+m-1] = \mathrm{X}[t, j-1] \circ 1) \le 1/2^m$. It would then follow from the union bound that $\Pr(\exists i \ne j \ : \ \mathrm{X}[i, i+m-1] = \mathrm{X}[j-m+1, j-1 \ \circ 1]) \le (n-m)/2^m \le \delta$.

Fix any $i \ne t$. Suppose, for example, $i < t$ (the case $i > t$ is dealt with similarly). $\mathrm{X}[i, i+m-1] = \mathrm{X}[t, j-1] \circ 1$ if and only if $\mathrm{X}[i, i+m-2] = \mathrm{X}[t, j-1]$ and $\mathrm{X}_{i+m-1} = 1$. It is easy to verify that the two events are independent and that the probability of the second is $1/2$. Thus, it suffices to show $\Pr(\mathrm{X}[i, i+m-2] = \mathrm{X}[t, j-1]) = 1/2^{m-1}$.

We will denote $\mathrm{X}^i = \mathrm{X}[i, i+m-2]$ and $\mathrm{X}^t = \mathrm{X}[t, j-1]$. Let p be the length of the longest prefix of X^i that does not overlap X^t (p can be anywhere between 1 and $m-1$). Divide X^i and X^t into $s \stackrel{\text{def}}{=} \lceil (m-1)/p \rceil$ non-overlapping blocks of size p each (except, maybe, for the last one which is shorter). Call the corresponding substrings $\mathrm{X}^i_1, \ldots, \mathrm{X}^i_s$ and $\mathrm{X}^t_1, \ldots, \mathrm{X}^t_s$, respectively. Note that $\mathrm{X}^i_q = \mathrm{X}^t_{q-1}$ for $q = 2, \ldots, s$. $\mathrm{X}^t = \mathrm{X}^i$ if and only if $\mathrm{X}^t_q = \mathrm{X}^i_q$ for all $q = 1, \ldots, s$, or equivalently, if and only if all the s blocks of X^t equal to X^i_1 (except, maybe, the last one which needs to be a prefix of X^i_1). The latter event occurs with probability $1/2^{m-1}$, since the X^t and X^i_1 are disjoint substrings of X, and the bits of X are chosen independently at random.

Recall that we assumed the probability Π errs on $(\mathrm{x}, (i, \mathrm{y}))$ is at most δ. Using Proposition 3 and applying a union bound, we get that the error probability of Π' is at most 2δ, completing the proof of Lemma 1.

Next, we obtain a lower bound on the one-way communication complexity of the modified indexing function. The proof is based on information-theoretic arguments.

Lemma 2. *Any one-way deterministic protocol that computes* $\mathrm{IND}_{n,k}$ *with error probability at most* ε *on inputs chosen according to* ν *requires at least* $(n-k)(1 - H_2(\varepsilon))$ *bits of communication.*

Proof. Fix any such deterministic protocol, and let $A(\cdot)$ denote the function Alice applies on her input in this protocol to determine the message she sends to Bob. Bob outputs an answer based on the message from Alice and based on his input. Thus, using the random variables $A(\mathrm{X})$, J, and $\mathrm{X}_{J-k}, \ldots, \mathrm{X}_{J-1}$, Bob is able to predict the random variable X_J with probability of error at most ε. This is exactly the scenario captured by a classical result from information theory, called *Fano's inequality* (cf. [10]), which implies $H_2(\varepsilon) \ \ \ge \ \ \mathrm{H}(\mathrm{X}_J \mid$

$A(\mathrm{X}), J, \mathrm{X}_{J-k}, \ldots, \mathrm{X}_{J-1})$. Here $\mathrm{H}(Z \mid Y)$ denotes the conditional Shannon entropy of the random variable Z given the random variable Y (cf. [10]). By definition, the conditional entropy $\mathrm{H}(Z \mid Y)$ equals to $\sum_y \mathrm{H}(Z \mid Y = y) \cdot \Pr(Y = y)$, where $\mathrm{H}(Z \mid Y = y)$ is the entropy of the conditional distribution of Z given the event $\{Y = y\}$. Expanding over the random variable J, we thus have:

$$\begin{aligned} H_2(\varepsilon) &\geq \frac{1}{n-k} \sum_{j=k+1}^{n} \mathrm{H}(\mathrm{X}_J \mid A(\mathrm{X}), \mathrm{X}_{J-k}, \ldots, \mathrm{X}_{J-1}, J = j) \\ &= \frac{1}{n-k} \sum_{j=k+1}^{n} \mathrm{H}(\mathrm{X}_j \mid A(\mathrm{X}), \mathrm{X}_{j-k}, \ldots, \mathrm{X}_{j-1}). \end{aligned}$$

Conditioning one variable on another can only reduce its entropy. Therefore, we can lower bound the j-th term on the righthand side by the conditional entropy $\mathrm{H}(\mathrm{X}_j \mid A(\mathrm{X}), \mathrm{X}_1, \ldots, \mathrm{X}_{j-1})$ (we added the random variables $\mathrm{X}_1, \ldots, \mathrm{X}_{j-k-1}$ to the conditioning). We thus have:

$$\begin{aligned} H_2(\varepsilon) &\geq \frac{1}{n-k} \sum_{j=k+1}^{n} \mathrm{H}(\mathrm{X}_j \mid A(\mathrm{X}), \mathrm{X}_1, \ldots, \mathrm{X}_{j-1}) \\ &\geq \frac{1}{n-k} \mathrm{H}(\mathrm{X}_{k+1}, \ldots, \mathrm{X}_n \mid A(\mathrm{X}), \mathrm{X}_1, \ldots, \mathrm{X}_k). \end{aligned}$$

The last transition follows from the chain rule for entropy. Another application of this rule implies that $\mathrm{H}(\mathrm{X}_{k+1}, \ldots, \mathrm{X}_n \mid A(\mathrm{X}), \mathrm{X}_1, \ldots, \mathrm{X}_k) = \mathrm{H}(\mathrm{X}_1, \ldots, \mathrm{X}_n, A(\mathrm{X})) - \mathrm{H}(A(\mathrm{X})) - \mathrm{H}(\mathrm{X}_1, \ldots, \mathrm{X}_k \mid A(\mathrm{X}))$. Since $A(\mathrm{X})$ fully depends on $\mathrm{X} = (\mathrm{X}_1, \ldots, \mathrm{X}_n)$, we have

$$\mathrm{H}(\mathrm{X}_1, \ldots, \mathrm{X}_n, A(\mathrm{X})) = \mathrm{H}(\mathrm{X}).$$

But, $\mathrm{H}(\mathrm{X}) = n$, as X has a uniform distribution on $\{0,1\}^n$. Since conditioning reduces entropy, $\mathrm{H}(\mathrm{X}_1, \ldots, \mathrm{X}_k \mid A(\mathrm{X})) \leq \mathrm{H}(\mathrm{X}_1, \ldots, \mathrm{X}_k) = k$. To conclude:

$$H_2(\varepsilon) \;\geq\; \frac{1}{n-k} \cdot (n - \mathrm{H}(A(\mathrm{X})) - k).$$

Therefore, $\mathrm{H}(A(\mathrm{X})) \geq (n-k)(1 - H_2(\varepsilon))$. Since $\mathrm{H}(A(\mathrm{X}))$ is always a lower bound on the length of $A(\mathrm{X})$, the lemma follows.

The proof of Theorem 1 now follows from Lemma 2, Lemma 1, and Yao's lemma [25].

For the case when $m \leq O(\log n)$, we have the following lower bound.

Theorem 2. *If $n \geq 2^{m/3} \cdot m$, then any δ-error randomized sketching algorithm for the pattern matching problem $\mathrm{PM}_{n,m}$ requires a sketch of size at least $2^{m/3-1} \cdot (1 - H_2(\delta))$.*

Proof. The lower bound follows from a reduction from the indexing problem, IND_t. The reduction works as follows. Let m be such that $t = 2^{m/3-1})$ and let $n \geq 2^{m/3} \cdot m$. Given her input $\mathrm{x} \in \{0,1\}^t$, Alice maps it first into a string $\mathrm{x}' \in \{0,1,\$\}^{n/3}$; \$ is a special symbol, which we call a "marker". Let $i_1, \ldots, i_k$ be the positions in which x has a 1. Then $\mathrm{x}' = i_1\$i_2\$\ldots i_k\$\$\ldots\$$ where each integer i_j is written in binary using $\log t$ bits and the trailing \$'s are used to make sure the string is of length $n/3$. Bob maps his input $j \in [t]$ into the string $\mathrm{y}' = j\$$ in $\{0,1,\$\}^{m/3}$. Note that since a marker can match only a marker, $\mathrm{x}_j = 1$ if and only if the pattern y′ matches x′.

In order to complete the reduction, we need to describe a mapping ϕ from strings over the ternary alphabet into (longer) strings over a binary alphabet, so that a pattern y′ matches a substring of x′ over the ternary alphabet if and only if $\phi(\mathrm{y}')$ matches a substring of $\phi(\mathrm{x}')$. ϕ could be, for example, the following mapping: 0 maps to 010, 1 maps to 101, and \$ maps to 111.

4 Approximate pattern matching

In this section we give a nearly-tight sketching lower bound for approximate pattern matching. Once again we use the lower bound for the indexing function to obtain our lower bound. Recall that we denote by $\mathrm{APM}_{n,m,\varepsilon}$ the problem in which Alice gets a string $\mathrm{x} \in \{0,1\}^n$, Bob gets a string $\mathrm{y} \in \{0,1\}^m$, and their goal is to determine whether there exists an index $i \in [n]$ such that $\mathrm{x}[i, i+m-1] = \mathrm{y}$, or whether for all $i \in [n]$, $\mathrm{HD}(\mathrm{x}[i, i+m-1], \mathrm{y}) \geq \varepsilon m$, assuming that one of the two holds.

Theorem 3. *If $n \leq 2^{O(m)}$, then for any constant $0 < \varepsilon < 1/8$, any randomized sketching algorithm for $\mathrm{APM}_{n,m,\varepsilon}$ requires a sketch of size $\Omega(\frac{n}{m})$.*

Proof. The proof works by a reduction from the indexing function IND_t; we assume, for simplicity, m divides n, and let $t = n/m$. We first need to fix a collection $\mathrm{z}_1, \ldots, \mathrm{z}_{2t}$ of $2t$ binary strings in $\{0,1\}^m$ with the following property: for any $m/2 \leq s \leq m$, and for any i, j, the prefix of z_i of length s and the suffix of z_j of length s have a large Hamming distance, namely, $\mathrm{HD}(\mathrm{z}_i[1,s], \mathrm{z}_j[m-s+1, m]) \geq \frac{m}{8}$. A simple probabilistic argument can show that such a collection exists as long as $t \leq 2^{\gamma m}$, for some constant $0 < \gamma < 1$.[2] We will call the last t strings in the collection also $\mathrm{o}_1, \ldots \mathrm{o}_t$.

Suppose Π is a sketching algorithm for $\mathrm{APM}_{n,m,\varepsilon}$. We use it to construct a sketching algorithm Π' for IND_t. Given her indexing input $\mathrm{x} \in \{0,1\}^t$, Alice maps it into a string $\mathrm{u} \in \{0,1\}^n$, which is formed by concatenating t strings of length m each. The j-th string is z_j if $\mathrm{x}_j = 0$ and it is o_j if $\mathrm{x}_j = 1$. Bob maps his input $i \in [t]$ into the string $\mathrm{v} = \mathrm{o}_i$.

Alice and Bob now run the algorithm Π on the inputs u and v. Bob decides that $\mathrm{x}_i = 1$ if and only if it is determined that v approximately matches u.

If $\mathrm{x}_i = 1$, then $\mathrm{v} = \mathrm{o}_i$ is a substring of u, and therefore the algorithm will be correct with high probability. If $\mathrm{x}_i = 0$, then v is not one of the strings

[2] Note that the corresponding Hamming distance is the summation of indicators for the events $(\mathrm{z}_i)_\ell = (\mathrm{z}_j)_{\ell+m-s}$ taken over $\ell = 1, \ldots, s$, and even if $i = j$, at least $s/2 \geq m/2$ of them are *independent*.

constituting u. We need to use now the property of the collection $z_1, \ldots, z_{2t}$ to show that no substring of u of length m has small Hamming distance from v.

Let $u^1, \ldots, u^t$ be the strings (which are taken from $z_1, \ldots, z_{2t}$) that constitute u. Suppose, to the contradiction, u has a substring α of length m such that $\mathrm{HD}(\alpha, v) \leq \varepsilon m \leq m/8$. The prefix of α overlaps some u^j and its suffix overlaps u^{j+1}. Call the overlapping prefix α_1 and the overlapping suffix α_2. At least one of α_1, α_2 has to be of size at least $m/2$. Suppose, for example, it is α_1, and let $s = |\alpha_1|$. Let v_1 be the prefix of v of length s. Since the total Hamming distance between α and v is at most $m/8$, also the Hamming distance between α_1 and v_1 is at most $m/8$. But that implies that the Hamming distance between the last s bits of u^j and the first s bits of v is at most $m/8$ even though $u^j \neq v$. This is a contradiction to the property of the collection $z_1, \ldots, z_{2t}$. We conclude that Π' is correct with high probability also when $x_i = 0$.

The lower bound now follows from the $\Omega(t)$ lower bound for indexing.

Theorem 4. *For any constant $\varepsilon > 0$, there is a randomized sketching algorithm for $\mathrm{APM}_{n,m,\varepsilon}$ with sketch size $O(\frac{n}{m}\varepsilon^{-1} \log n)$.*

Proof. We may assume that the text size is at most $n' = (1+\varepsilon)m$, because Alice can divide the text x into substrings of length $(1+\varepsilon)m$ having an overlap of m bits between successive substrings, and then Alice and Bob apply an $O(\frac{n'}{m} \log n)$ sketch independently for of these substrings. If the pattern matches the text, then at least one of these substrings must contain that pattern. If the text does not contain the pattern, then it suffices to have the algorithm err with probability at most $1/n^2$ on each of the $\frac{n}{\varepsilon m}$ substrings.

If the text size is $n \leq (1+\varepsilon)m$, Alice simply computes $O(\log n)$ random inner products $\sum_j x_j r_j \pmod 2$ à la Kushilevitz, Ostrovsky and Rabani [19], and sends them to Bob. These inner products are tuned to determine whether the Hamming distance is at most $\varepsilon m/2$ or whether it is at least εm, namely, each bit r_j is chosen independently to be 1 with probability $1/(2\varepsilon m)$ and 0 with probability $1 - 1/(2\varepsilon m)$. Since each inner product results in a single bit, the sketch size is clearly $O(\log n)$.

It remains to show how Bob uses the results of these $O(\log n)$ inner products to determine, with high probability, whether $x[i, i+m-1] = y$. Notice that in each inner product,

$$\Pr[r_j = 0 \text{ for all } j < i \text{ and for all } j \geq i+m] = (1 - 1/(2\varepsilon m))^{\varepsilon m} = \Omega(1).$$

That is, with probability at least some constant, any single inner product with x is actually also a random inner product with $x[i, i+m-1]$, and hence *can be used* by Bob to estimate whether the Hamming distance $\mathrm{HD}(x[i, i+m-1], y)$ is at most $\varepsilon m/2$ or at least εm. The details of this estimate are exactly as in [19]; in short, in the latter case the probability that the inner product turns out to be 1 is higher additively by a constant than in the former. By a standard Chernoff bound, with high probability (say at least $1 - 1/n^3$), there are some $\Omega(\log n)$ inner products that Bob can use to estimate $\mathrm{HD}(x[i, i+m-1], y)$. The proof now follows by a union bound over the $n' \leq n$ possible values of i.

Remark. The above sketching algorithm is actually stronger than claimed in the theorem, as it determines, with high probability, whether there exists an index $i \in [n]$ such that $\mathrm{HD}(\mathrm{x}[i, i+m-1], \mathrm{y}) \leq \varepsilon m/2$, or whether for all $i \in [n]$, $\mathrm{HD}(\mathrm{x}[i, i+m-1], \mathrm{y}) \geq \varepsilon m$, assuming that one of the two holds.

References

1. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
2. A. Amir and G. Benson. Efficient two-dimensional compressed matching. In *Proceedings of IEEE Data Compression Conference (DCC)*, pages 279–288, 1992.
3. A. Amir, G. Benson, and M. Farach. Let sleeping files lie: Pattern matching in Z-compressed files. *J. of Computer and System Sciences*, 52(2):299–307, 1996.
4. Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. Manuscript, 2004.
5. Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. Information theory methods in communication complexity. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, pages 93–102, 2002.
6. T. Batu, F. Ergün, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 316–324, 2003.
7. A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.
8. A. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *WWW6/Computer Networks*, 29(8–13):1157–1166, 1997.
9. M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.
10. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.
11. E. de Moura, G. Navarro, N. Ziviani, and R. Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, 18(2):113–139, 2000.
12. M. Farach and M. Thorup. String matching in Lempel-Ziv compressed strings. *Algorithmica*, 20(4):388–404, 1998.
13. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In *28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 927–938. Springer, 2001.
14. J. Feigenbaum, S. Kannan, M. J. Strauss, and M. Viswanathan. An approximate L^1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002/03.
15. P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 390–398. IEEE Computer Society, 2000.
16. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

17. R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.
18. I. Kremer, N. Nisan, and D. Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.
19. E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.
20. S. Lonardi. Pattern matching pointers. *Available* *http://www.cs.ucr.edu/~stelo/pattern.html*, 2004.
21. U. Manber. A text compression scheme that allows fast searching directly in the compressed file. *ACM Transactions on Information Systems*, 15(2):124–136, 1997.
22. G. Navarro and J. Tarhio. Boyer-Moore string matching over Ziv-Lempel compressed text. In *Proceedings of 11th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 1848 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2000.
23. I. Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.
24. Y. Shibata, T. Matsumoto, M. Takeda, A. Shinohara, and S. Arikawa. A Boyer-Moore type algorithm for compressed pattern matching. In *Proceedings of 11th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 1848 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2000.
25. A. C.-C. Yao. Lower bounds by probabilistic arguments. In *Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science*, pages 420–428, 1983.