Recent Advances in Text-to-Pattern Distance Algorithms

Przemysław Uznański^(✉)

Institute of Computer Science, University of Wrocław, Wrocław, Poland
puznanski@cs.uni.wroc.pl

Abstract. Computing text-to-pattern distances is a fundamental problem in pattern matching. Given a text of length n and a pattern of length m, we are asked to output the distance between the pattern and every n-substring of the text. A basic variant of this problem is computation of Hamming distances, that is counting the number of mismatches (different characters aligned), for each alignment. Other popular variants include ℓ_1 distance (Manhattan distance), ℓ_2 distance (Euclidean distance) and general ℓ_p distance. While each of those problems trivially generalizes classical pattern-matching, the efficient algorithms for them require a broader set of tools, usually involving both algebraic and combinatorial insights. We briefly survey the history of the problems, and then focus on the progress made in the past few years in many specific settings: fine-grained complexity and lower-bounds, $(1 + \varepsilon)$ multiplicative approximations, k-bounded relaxations, streaming algorithms, purely combinatorial algorithms, and other recently proposed variants.

1 Hamming Distance

A most fundamental problem in stringology is that of pattern matching: given pattern P and text T, find all occurrences of P in T where by occurrence we mean a substring (a consecutive fragment) of T that is identical to P. A huge efforts have been put into advancement of understanding of pattern matching by the community. One particular variant to consider is finding occurrences or almost-occurrences of P in T. For this, we need to specify almost-occurrences: e.g. introduce some form of measure of distance between words, and then look for substrings of T which are close to P. We are interested in measures that are position-based, that is they are defined over strings of equal length, and are based upon distances between letters on corresponding positions (thus e.g. edit distance is out of scope of this survey). Consider for example

Definition 1 (Hamming distance). *For strings A, B of equal length, their Hamming distance is defined as*

$$Ham(A, B) = |\{i : A[i] \neq B[i]\}|.$$

Supported by Polish National Science Centre grant 2019/33/B/ST6/00298.

Hence, the Hamming distance counts the number of *mismatches* between two words. This leads us to the core problem considered in this survey.

Definition 2 (Text-to-pattern Hamming distance). *For a text $T[1,n]$ and a pattern $P[1,m]$, the text-to-pattern Hamming distance asks for an output array $O[1,n-m+1]$ such that*

$$O[i] = Ham(P, T[i, i+m-1]).$$

Observe that this problem generalizes the detection of almost-occurrences – one can scan the output array and output positions with small distance to the pattern.

1.1 Convolution in Text-to-Pattern Distance

Convolution of two vectors (arrays) is defined as follow

Definition 3 (Convolution). *For 0-based vectors A and B we define their convolution $(A \circ B)$ as a vector:*

$$(A \circ B)[k] = \sum_{i+j=k} A[i] \cdot B[j].$$

Such a definition has a natural interpretation e.g. in terms of polynomial product: if we interpret a vector as coefficients of a polynomial that is $A(x) = \sum_i A[i] \cdot x^i$ and $B(x) = \sum_i B[i] \cdot x^i$, then $(A \circ B)$ are coefficients of $A(x) \cdot B(x)$.

Convolution over integers is computed by Fast Fourier transform (FFT) in time $\mathcal{O}(n \log n)$. This requires actual embedding of integers into field, e.g. $\mathbb{F}_p$ or $\mathbb{C}$. This comes at a cost, if e.g. we were to consider text-to-pattern distance over (non-integer) alphabets that admit only field operations, e.g. matrices or geometric points. Convolution can be computed using a "simpler" set of operations, that is just with ring operations in e.g. $\mathbb{Z}_p$ using Toom-Cook multiplication [35], which is a generalization of famous divide-and-conquer Karatsuba algorithm [20]. However, not using FFT makes the algorithm slower, with Toom-Cook algorithm taking time $\mathcal{O}(n 2^{\sqrt{2 \log n}} \log n)$, and increases the complexity of the algorithm.

Fischer and Paterson in [16] observed that convolution can be used to compute text-to-pattern Hamming distance for small alphabets. Consider the following observation: for binary P and T, denote by P' the *reversed* P. Then we have the following property:

$$\begin{aligned}
\mathrm{Ham}(P, T[i, i+m-1]) &= \sum_j \Big(T[i+j](1 - P[j]) + (1 - T[i+j])P[j] \Big) \\
&= \sum_{j+k=m+1+i} \Big(T[j]\overline{P'[k]} + \overline{T[j]}P'[k] \Big) \\
&= (T \circ \overline{P'})[m+1+i] + (\overline{T} \circ P')[m+1+i]
\end{aligned}$$

where e.g. $\overline{T}$ denotes *negating* every entry of T. Thus the whole algorithm is done by computing two convolutions in time $\mathcal{O}(n \log m)$.[1] This approach in fact generalizes to arbitrary size alphabets by following observation: "contribution" of single $c \in \Sigma$ to number of mismatches for all positions can be computed with single convolution. This results in $\mathcal{O}(|\Sigma| n \log m)$ time algorithm.

The natural question is whether faster (than naive quadratic-time) algorithms for large alphabets exist. The answer is affirmative, by (almost simultaneous) results of Abrahamson [1] and Kosaraju [24]. The insight is that for any letter $c \in |\Sigma|$, we can compute its "contribution" twofold:

- by FFT in time $\mathcal{O}(n \log m)$,
- or in time $\mathcal{O}(t)$ per each of n alignments, where t is the number of occurrences of c in lets say pattern.

The insight is that we apply the former for letters that appear often ("dense" case) and latter for sparse letters. Since there can be at most m/T letters that appear at least T times in pattern each, the total running time is $\mathcal{O}(n \log m \cdot \frac{m}{T} + Tn)$ which is minimized when $T = \sqrt{m \log m}$ with run-time $\mathcal{O}(n\sqrt{m \log m})$.

This form of mixing combinatorial and algebraical insights is typical for the type of problems considered in this paper, and we will see more of it in the following sections. As a side-note, the complexity of $\mathcal{O}(n\sqrt{m \log m})$ remains state-of-the-art.

1.2 Relaxation: k-Bounded Distances

The lack of progress in Hamming text-to-pattern distance complexity sparked interest in searching for relaxations of the problem, in hope of reaching linear (or almost linear) run-time. For example if we consider reporting only the values not exceeding a certain threshold value k, then we have the so-called k-approximated distance. The motivation comes from the fact that if we are looking for almost-occurrences, then if the distance is larger than a certain threshold value, the text fragment is too dissimilar to pattern and we are safe to discard it.

The very first solution to this problem was shown by Landau and Vishkin [26] working in time $\mathcal{O}(nk)$, using an essentially combinatorial approach of taking $\mathcal{O}(1)$ time per mismatch per alignment using LCP queries (Longest Common Prefix queries), where $\mathrm{LCP}(i, j)$ returns maximal k such that $T[i, i+k] = P[j, j+k]$. This solution requires preprocessing of T and P with e.g. suffix tree, which is a standard tool-set of stringology. This solution still is slower than naive algorithm for $k = m^{1/2+\delta}$, but has the nice property of using actually $\mathcal{O}(n)$ time for constant k. This technique is also known as *kangaroo jumps*.

This initiated a series of improvements to the complexity, with algorithms of complexity $\mathcal{O}(n\sqrt{k \log k})$ and $\mathcal{O}((k^3 \log k + m) \cdot n/m)$ by Amir et al. [4]. First algorithm is an adaptation of general algorithm of Abrahamson with balancing

[1] Its $\log m$ not $\log n$ by standard trick of reducing the problem to $\lceil n/m \rceil$ instances with pattern P of length m and text of length $2m$.

of "sparse" vs. "dense" case done w.r.t. k instead of m (some further combinatorial insights are required to make the cases work with proper run-time). Such trade-off has this nice property that for $k = m$ the complexity matches that of Abrahamson's algorithm. Second algorithm is more interesting, since it shows that for non-trivial values of k (in this case, $k = \mathcal{O}(m^{1/3})$) near-linear time algorithms are possible.

The later complexity was then improved to $\mathcal{O}((k^2 \log k + m \operatorname{poly}\log m) \cdot n/m)$ by Clifford et al. [13]. We now discuss the techniques of this algorithm, starting with *kernelization* technique.

Definition 4 ([13]). *An integer $\pi > 0$ is an x-period of a string $S[1, m]$, if Ham$(S[\pi + 1, m], S[1, m - \pi]) \le x$ and π is minimal such integer.*

Such definition should be compared with regular definition of a period, where π is a period of string S if $S[\pi + 1, m] = S[1, m - \pi]$.

We then observe the following:

Lemma 1 ([13]). *If ℓ is a $2x$-period of the pattern, then any two occurrences of the pattern in the text with at most x mismatches are at offset distance at least ℓ.*

The first step of the algorithm is to determine some small $\mathcal{O}(k)$-period of the pattern. This actually does not require any specialized machinery and can be done with a 2-approximate algorithm for text-to-pattern Hamming distance (multiplicative approximations are a topic of the following section). We then distinguish two cases, where small means $\mathcal{O}(k)$.

No small k-period. This is an "easy" case, where a filtering step allows us to keep only $\mathcal{O}(n/k)$ alignments that are candidates for $\le k$-distance matches. A "kangaroo jumps" technique of Landau and Vishkin allows us to verify each one of them in $\mathcal{O}(k)$ time, resulting in linear time spent in this case.

Small $2k$-period. This is a case where we can deduce some regularity properties. Denote the $2k$-period as ℓ. First, P can be decomposed into ℓ words from its arithmetic progressions of positions, with step ℓ and every possible offset. From the definition of ℓ being $2k$-period, we know that the total number of runs in those words is small. The more interesting property is that even though the text T can be arbitrary, if T is not regular enough it can be discarded (and this actually concerns any part of the text that is not regular). More precisely, there is a substring of text that is regular enough and contains all the alignments of P that are at Hamming distance at most k (assuming $n = 2m$, which we can always guarantee).
What remains is to observe that finding ℓ, compressing of P into arithmetic progressions and finding compressible region T' of T all can be done in $\widetilde{\mathcal{O}}(n)$ time, and that all of alignments of text to pattern correspond to alignments of those arithmetic progressions, and can be solved in $\mathcal{O}(k^2)$ time.

Final step in the sequence of improvements to this problem was done by Gawrychowski and Uznański [17]. They observe that the algorithm from [13] can be interpreted in terms of reduction: instance of k-bounded text-to-pattern Hamming distance with T and P is reduced to new T' and P', where T' and P' are possibly of the same length, but have total number of runs in their Run-length encoding (RLE) representation bounded as $\mathcal{O}(k)$. The algorithm from [13] then falls back to brute force $\mathcal{O}(k^2)$ time computation. While $\mathcal{O}(k^{2-\delta})$ algorithm for RLE-compressed pattern matching would falsify 3-SUM conjecture (c.f. [10]), some structural properties of the instances can be leveraged based on the fact that they are RLE-compressed from inputs of length m. A balancing argument (in style of one from [4] or [1]) follows, allowing to solve this sub-problem in time $\mathcal{O}(k\sqrt{m \log m})$. The final complexity for the whole algorithm becomes then $\widetilde{\mathcal{O}}((m + k\sqrt{m}) \cdot n/m)$.

1.3 Relaxation: $1 \pm \varepsilon$ Approximation

Another way to relax to text-to-pattern distance is to consider multiplicative approximation when reporting number of mismatches. The very elegant argument made by Karloff [21] states the following.

Observation 1. *Consider a randomly chosen projection $\varphi : \Sigma \to \{0,1\}$ (each letters mapping is chosen independently and uniformly at random) and words A, B. Then*

$$\mathbb{E}[Ham(\varphi(A), \varphi(B))] = \frac{1}{2} Ham(A, B),$$

where $\varphi(A)$ denotes applying φ to each letter of A separately.

Thus the algorithm consists of: (i) choosing independently at random K random projections; (ii) for each projection, computing text-to-pattern Hamming distance over projected input; (iii) averaging answers. A concentration argument then follows, giving standard $K = \mathcal{O}(\frac{\log n}{\varepsilon^2})$ independent repetitions guaranteeing that average recovers actual Hamming distance with $(1 \pm \varepsilon)$ multiplicative guarantee, with high probability. This gives total run-time $\widetilde{\mathcal{O}}(n/\varepsilon^2)$.

The $\frac{1}{\varepsilon^2}$ dependency was believed to be inherent, as is the case for e.g. space complexity of sketching of Hamming distance, cf. [8,19,38]. However, for approximate pattern matching that was refuted in Kopelowitz and Porat [22,23], where randomized algorithms were provided with complexity $\mathcal{O}(\frac{n}{\varepsilon} \log n \log m \log \frac{1}{\varepsilon} \log |\Sigma|)$ and $\mathcal{O}(\frac{n}{\varepsilon} \log n \log m)$ respectively. The second mentioned algorithm is actually surprisingly simple: instead of projecting onto binary alphabet, random projections $\Sigma \to [u]$ are used, where $u = \mathcal{O}(\varepsilon^{-1})$. Such projections collapse in expectation only an ε-fraction of mismatches, introducing systematic $1 + \varepsilon$ multiplicative error. A simple Markov bound argument follows, that since *expected* error is within desired bound, taking few (lets say $\log n$) repetitions and taking median guarantees recovery of good approximate answer with high probability. What remains to observe is that exact counting of text-to-pattern distance over projected alphabet takes u repetitions of convolution, so the total runtime is $\widetilde{\mathcal{O}}(n/\varepsilon)$. An alternative exposition to this result was provided in [34].

2 Other Norms

A natural extension to counting mismatches is to consider other norms (e.g. ℓ_1, ℓ_2, general ℓ_p norm or ℓ_∞ norm), or to move beyond norms (so called *threshold* pattern matching c.f. Atallah and Duket [6] or dominance pattern matching c.f. Amir and Farach [3]).

Definition 5 (ℓ_p distance). *For two strings of equal length over integer alphabet and constant $p > 0$, their ℓ_p distance is defined as*

$$\|A - B\|_p = \Big(\sum_i \Big|A[i] - B[i]\Big|^p\Big)^{1/p}.$$

Definition 6 (ℓ_∞ distance). *For two strings of equal length over integer alphabet, their ℓ_∞ distance is defined as*

$$\|A - B\|_\infty = \max_i \Big|A[i] - B[i]\Big|.$$

2.1 Exact Algorithms

To see that the link between convolution and text-to-pattern distance is relevant when considering other norms, consider the case of computing ℓ_2 distances. We are computing output array $O[]$ such that $O[i] = \sum_j (T[i+j] - P[j])^2$. However, this is equivalent to computing, for every i simultaneously, value $\sum_j T[i+j]^2 + \sum_j P[j]^2 - 2\sum_j T[i+j]P[j]$. While the terms $\sum_j T[i+j]^2$ and $\sum_j P[j]^2$ can be easily precomputed in $\mathcal{O}(n)$ time, we observe (following [29]) that $\sum_j T[i+j]P[j]$ is essentially convolution. Indeed, consider P' such that $P'[j] = P[m+1-j]$, and then what follows.

We now consider ℓ_1 distance. Using techniques similar to Hamming distance, the $\mathcal{O}(n\sqrt{n \log n})$ complexity algorithms were developed independently in 2005 by Clifford et al. [11] and Amir et al. [5] for reporting all ℓ_1 distances. The algorithms use a balancing argument, starting with observation that alphabet can be partitioned into buckets, where each bucket is a consecutive interval of alphabet. The contribution of characters from the same interval is counted in one phase, and contribution of characters from distinct intervals is counted in second phase.

Interestingly, no known algorithm for *exact* computation of text-to-pattern ℓ_p distance for arbitrary value of p is known. By the folklore observation, for any even p we can reduce it to convolution and have $\mathcal{O}(n \log m)$ time algorithm (c.f. Lipsky and Porat [29], with $\mathcal{O}$ hiding p^2 dependency). By the results of Labib et al. [25] any odd-value integer p admits $\widetilde{\mathcal{O}}(n\sqrt{m \log m})$ time algorithm (the algorithm is given implicitly, by providing a reduction from ℓ_p to Hamming distance, with $\widetilde{\mathcal{O}}$ hiding $(\log m)^{\mathcal{O}(p)}$ dependency).

2.2 Approximate and k-Bounded Algorithms

Once again, the topic spurs interest in approximation algorithm for distance functions. In [29] a deterministic algorithm with a run time of $\mathcal{O}(\frac{n}{\varepsilon^2} \log m \log U)$ was given, while later in [17] the complexity has been improved to a (randomized) $\mathcal{O}(\frac{n}{\varepsilon} \log^2 n \log m \log U)$, where U is the maximal integer value on the input. Later [34] it was shown that such complexity is in fact achievable (up to poly-log factors) with a deterministic solution. All those solutions follow similar framework of *linearity-preserving* reductions, which has actually broader applications. The framework is as follow: imagine we want to approximate some distance function $d : \Sigma \times \Sigma \to \mathbb{R}$. We build small number of pairs of projections, $L_1, \ldots, L_t, R_1, \ldots, R_t$, with the following property: $d(x, y) \approx \sum_i L_i(x) \cdot R_i(x)$.[2] Given such formulation, by linearity, text-to-pattern of A and B using distance function d is approximated by a linear combination of convolutions of $L_i(A)$ and $R_i(B)$. The complexity of the solutions follows from the number of different projections that need to be used.

For ℓ_∞ distances, in [29] a $\widetilde{\mathcal{O}}(n/\varepsilon)$ time approximate solution was given, while in Lipsky and Porat [27] a k-bounded ℓ_∞ distance algorithm with time $\mathcal{O}(nk \log m)$ was given. For k-bounded ℓ_1 distances, [5] a $\mathcal{O}(n\sqrt{k \log k})$ run-time algorithm was given, while in [17] an algorithm with run-time $\widetilde{\mathcal{O}}((m + k\sqrt{m}) \cdot n/m)$ was given. The fact that those run-times are (up to poly-logs) identical to corresponding run-times of k-bounded Hamming distances is not a coincidence, as [17] have shown that k-bounded ℓ_1 is at least as easy as k-bounded Hamming distance reporting.

A folklore result (c.f. [29]) states that the randomized algorithm with a run time of $\widetilde{\mathcal{O}}(\frac{n}{\varepsilon^2})$ is in fact possible for any ℓ_p distance, $0 < p \leq 2$, with use of p-stable distributions and convolution. Such distributions exist only when $p \leq 2$, which puts a limit on this approach. See [30] for wider discussion on p-stable distributions. Porat and Efremenko [32] has shown how to approximate general distance functions between pattern and text in time $\widetilde{\mathcal{O}}(\frac{n}{\varepsilon^2})$. Their solution does not immediately translates to ℓ_p distances, since it allows only for score functions of form $\sum_j d(t_{i+j}, p_j)$ where d is arbitrary metric over Σ. Authors state that their techniques generalize to computation of ℓ_2 distances, and in fact those generalize further to ℓ_p distances as well, but the ε^{-2} dependency in their approach is unavoidable. Finally, for any $p > 0$ there is ℓ_p distance $(1 \pm \varepsilon)$-approximate algorithm running in time $\widetilde{\mathcal{O}}(n/\varepsilon)$ by results shown in [34]. Final result follows the framework of linearity-preserving reductions.

3 Lower Bounds

It is a major open problem whether near-linear time algorithm, or even $\mathcal{O}(n^{3/2-\delta})$ time algorithms, are possible for such problems. A conditional lower bound was shown in [12], via a reduction from matrix multiplication. This means

[2] Here we used $\approx$ since its in the context of approximate algorithms. The same framework applies to exact algorithms, then we replace $\approx$ with $=$.

that existence of combinatorial algorithm with run-time $\mathcal{O}(n^{3/2-\delta})$ solving the problem for Hamming distances implies combinatorial algorithms for Boolean matrix multiplication with $\mathcal{O}(n^{3-\delta})$ run-time, which existence is unlikely. Looking for unconditional bounds, we can state this as a lower-bound of $\Omega(n^{\omega/2})$ for Hamming distances pattern matching, where $2 \leq \omega < 2.373$ is a matrix multiplication exponent. In fact those techniques can be generalized to take into account k-bounded version of this problem:

Theorem 2 **([17]).** *For any positive $\varepsilon, \alpha, \kappa$ such that $\frac{1}{2}\alpha \leq \kappa \leq \alpha \leq 1$ there is no combinatorial algorithm solving pattern matching with $k = \Theta(n^\kappa)$ mismatches in time $\mathcal{O}((k\sqrt{m})^{1-\varepsilon}) \cdot n/m)$ for a text of length n and a pattern of length $m = \Theta(n^\alpha)$, unless the combinatorial matrix multiplication conjecture fails.*

Complexity of pattern matching under Hamming distance and under ℓ_1 distance was proven to be identical (up to poly-logarithmic terms) in [25]. This equivalence in fact applies to a wider range of distance functions and in general other score functions. The result shows that a wide class of functions are equivalent under linearity-preserving reductions to computation of Hamming distances. The class includes e.g. dominance score, ℓ_1 distance, threshold score, ℓ_{2p+1} distance, any of above with wildcards, and in fact a wider class called piece-wise polynomial functions.

Definition 7. *For integers A, B, C and polynomial $P(x, y)$ we say that the function $P(x, y) \cdot \mathbb{1}[Ax + By + C > 0]$ is* half-plane polynomial. *We call a sum of half-plane polynomial functions a* piece-wise polynomial. *We say that a function is* axis-orthogonal piece-wise polynomial, *if it is piece-wise polynomial and for every i, $A_i = 0$ or $B_i = 0$.*

Observe that $\mathrm{Ham}(x, y) = \mathbb{1}[x > y] + \mathbb{1}[x < y]$, $\max(x, y) = x \cdot \mathbb{1}[x \geq y] + y \cdot \mathbb{1}[x < y]$, $|x - y|^{2p+1} = (x - y)^{2p+1} \cdot \mathbb{1}[x > y] + (y - x)^{2p+1} \cdot \mathbb{1}[x < y]$, and e.g. threshold function can be defined as $\mathrm{thr}_\delta(x, y) \overset{\text{def}}{=} \mathbb{1}[|x - y| \geq \delta] = \mathbb{1}[x \leq y - \delta] + \mathbb{1}[x \geq y + \delta]$.

Theorem 3. *Let $\diamond$ be a piece-wise polynomial of constant degree and poly $\log n$ number of summands.*

- *If $\diamond$ is axis orthogonal, then $\diamond$ is "easy": $(+, \diamond)$ convolution takes $\widetilde{O}(n)$ time, $(+, \diamond)$ matrix multiplication takes $\widetilde{O}(n^\omega)$ time.*
- *Otherwise, $\diamond$ is* Hamming distance complete*: under one-to-polylog reductions, on inputs bounded in absolute value by poly(n), $(+, \diamond)$ product is equivalent to Hamming distance, $(+, \diamond)$ convolution is equivalent to text-to-pattern Hamming distance and $(+, \diamond)$ matrix product is equivalent to Hamming-distance matrix product.*

Some of those reduction (for specific problems) were presented in literature, c.f. [28,37,39], but never as a generic class-of-problems equivalence.

This means that the encountered barrier for all of the induced text-to-pattern distance problems is in fact the same barrier, and we should not expect algorithms with dependency $n^{3/2-\delta}$ without some major breakthrough. Unfortunately such reductions do not preserve properties of k-bounded instances or $1 \pm \varepsilon$-approximate ones, so this result tells us nothing about relative complexity of relaxed problems, and it is a major open problem to do so.

4 Streaming Algorithms

In streaming algorithms, the goal is to process text in a streaming fashion, and answer in a real-time about the distance between last m characters of text and a pattern. The primary measure of efficiency is the memory complexity of the algorithm, that is we assume that the whole input (or even the whole pattern) is too large to fit into the memory and some for of small-space representation is required. The time to process each character is the secondary measure of efficiency, since it usually is linked to memory efficiency. By folklore result, exact reporting of e.g. Hamming distances is impossible in $o(m)$ memory, so the focus of the research has been on relaxed problems, that is k-bounded and $(1 \pm \varepsilon)$-approximate reporting.

For k-bounded reporting of Hamming distances, in Porat and Porat [31] a $\mathcal{O}(k^3)$ space and $\mathcal{O}(k^2)$ time per character streaming algorithm was presented. It was later improved in [13] to $\widetilde{\mathcal{O}}(k^2)$ space and $\widetilde{\mathcal{O}}(\sqrt{k})$ time per character, and then in Clifford et al. [14] to $\widetilde{\mathcal{O}}(k)$ space keeping $\widetilde{\mathcal{O}}(\sqrt{k})$ time per character. Many interesting techniques were developed for this problem. As an example, k-mismatch problem can be reduced to (k^2 many instances of) 1-mismatch problem (c.f. [13]), which in fact reduces to exact pattern matching in streaming model (c.f. [31]). Other approach is to construct efficient rolling sketches for k-mismatch problem, based on Reed-Solomon error correcting codes (c.f. [14]).

For $1 \pm \varepsilon$, two interesting approaches are possible. First approach was presented by Clifford and Starikovskaya [15] and later refined in Svagerka et al. [33]. This approach consists of using rolling sketches of text started every $\sim \sqrt{m}$ positions, and additionally $\sim \sqrt{m}$ sketches of substrings of length $m - \sqrt{m}$ of pattern are maintained (guaranteeing that at least one sketch in text is aligned to one sketch of long pattern fragment). One way of building rolling sketches for approximate Hamming distance is to use random projections to binary alphabet and reduce the problem to one for binary alphabet, where binary alphabet uses Johnson-Lindenstrauss type of constructions. This approach results in $\widetilde{\mathcal{O}}(\sqrt{m}/\varepsilon^2)$ memory and $\widetilde{\mathcal{O}}(1/\varepsilon^2)$ time per character.

Alternative approach was proposed in recent work of Chan et al. [9]. They start with observation that the Hamming distance can be estimated by checking mismatches at a random subset of positions. Their algorithm uses a random subset as follow: the algorithm picks a random prime p (of an appropriately chosen size) and a random offset b, and considers a subset of positions $\{b, b+p, b+2p, \ldots\}$. The structured nature of the subset enables more efficient computation. It turns out that even better efficiency is achieved by using multiple (but still relatively few) offsets. When approximating the Hamming distance of the pattern at subsequent text locations, the set of sampled positions in the text changes, and so a straightforward implementation seems too costly. To overcome this challenge, a key idea is to shift the sample a few times in the pattern and a few times in the text (namely, for a trade-off parameter z, our algorithm considers z shifts in the pattern and p/z shifts in the text). Interestingly, the proposed solution is even more efficient when considering a $(1\pm\varepsilon)$-approximate k-bounded reporting of Hamming distances.

Theorem 4 **([9]).** *There is an algorithm that reports $(1 \pm \varepsilon)$-approximate k-bounded Hamming distances in a streaming setting that uses $\widetilde{\mathcal{O}}(\min(\sqrt{k}/\varepsilon^2, \sqrt{m}/\varepsilon^{1.5}))$ space and takes $\mathcal{O}(1/\varepsilon^3)$ time per character.*

Focusing on other norms, we note that in [33] a sublinear space algorithms for ℓ_p norms for $0 < p \leq 2$ was presented. The specific details of construction vary between different values of p, and the techniques include: using p-stable distributions (c.f. [18]), range-summable hash functions (c.f. [7]) and Johnson-Lindenstrauss projections (c.f. [2]).

Theorem 5 **([33]).** *Let $\sigma = n^{\mathcal{O}(1)}$ denote size of alphabet. There is a streaming algorithm that computes a $(1 \pm \varepsilon)$-approximation of the ℓ_p distances. The parameters of the algorithm are*

1. *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log\sigma)$ space, and $\widetilde{\mathcal{O}}(\varepsilon^{-2})$ time per arrival when $p = 0$ (Hamming distance);*
2. *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log^2\sigma)$ space and $\widetilde{\mathcal{O}}(\sqrt{n}\log\sigma)$ time per arrival when $p = 1$;*
3. *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n}+\log^2\sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n})$ time per arrival when $0 < p < 1/2$;*
4. *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log^2\sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-3}\sqrt{n})$ time per arrival when $p = 1/2$;*
5. *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n}+\log^2\sigma)$ space and $\widetilde{\mathcal{O}}(\sigma^{\frac{2p-1}{1-p}}\sqrt{n}/\varepsilon^{2+3\cdot\frac{2p-1}{1-p}})$ time per arrival when $1/2 < p < 1$;*
6. *in $\widetilde{\mathcal{O}}(\varepsilon^{-2-p/2}\sqrt{n}\log^2\sigma)$ space and $\mathcal{O}(\varepsilon^{-p/2}\sqrt{n} + \varepsilon^{-2}\log\sigma)$ time per arrival for $1 < p \leq 2$.*

5 Open Problems

Below we list several open problems of the area, which we believe are the most promising research directions and/or pressing questions.

1. Show deterministic algorithm for $(1 \pm \varepsilon)$-approximate ℓ_p reporting for $0 < p < 1$, preferably in time $\widetilde{\mathcal{O}}(n/\varepsilon)$.
2. What is the time complexity of exact ℓ_p reporting for non-integer p?
3. Show conditional lower bound for exact Hamming distance reporting from stronger hypotheses, like 3SUM-HARDNESS.
4. Lower bounds for $1 \pm \varepsilon$ approximations (conditional between problems, or from external problems), for any of the discussed problems.
5. What is the true space complexity dependency in streaming $(1 \pm \varepsilon)$ approximate Hamming distance reporting? Is $\sqrt{m}\varepsilon^{-1.5}$ complexity optimal?

6. Can we close the gap between streaming complexity of approximate ℓ_p algorithms and streaming complexity of approximate Hamming distance?
7. Can we design effective "combinatorial" algorithms for all mentioned problems (e.g. not relying on convolution)? For Hamming, ℓ_1 and ℓ_2 distances answer is at least partially yes (c.f. [9] and [36]).

References

1. Abrahamson, K.R.: Generalized string matching. SIAM J. Comput. **16**(6), 1039–1051 (1987)
2. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J. Comput. Syst. Sci. **66**(4), 671–687 (2003). https://doi.org/10.1016/S0022-0000(03)00025-4
3. Amir, A., Farach, M.: Efficient matching of nonrectangular shapes. Ann. Math. Artif. Intell. **4**(3), 211–224 (1991). https://doi.org/10.1007/BF01531057
4. Amir, A., Lewenstein, M., Porat, E.: Faster algorithms for string matching with k mismatches. J. Algorithms **50**(2), 257–275 (2004). https://doi.org/10.1016/S0196-6774(03)00097-X
5. Amir, A., Lipsky, O., Porat, E., Umanski, J.: Approximate Matching in the L_1 Metric. In: Apostolico, A., Crochemore, M., Park, K. (eds.) CPM 2005. LNCS, vol. 3537, pp. 91–103. Springer, Heidelberg (2005). https://doi.org/10.1007/11496656_9
6. Atallah, M.J., Duket, T.W.: Pattern matching in the hamming distance with thresholds. Inf. Process. Lett. **111**(14), 674–677 (2011). https://doi.org/10.1016/j.ipl.2011.04.004
7. Calderbank, A.R., Gilbert, A.C., Levchenko, K., Muthukrishnan, S., Strauss, M.: Improved range-summable random variable construction algorithms. In: SODA, pp. 840–849 (2005)
8. Chakrabarti, A., Regev, O.: An optimal lower bound on the communication complexity of gap-hamming-distance. SIAM J. Comput. **41**(5), 1299–1317 (2012). https://doi.org/10.1137/120861072
9. Chan, T.M., Golan, S., Kociumaka, T., Kopelowitz, T., Porat, E.: Approximating text-to-pattern hamming distances. In: STOC 2020 (2020)
10. Chen, K.-Y., Hsu, P.-H., Chao, K.-M.: Approximate matching for run-length encoded strings is 3SUM-hard. In: Kucherov, G., Ukkonen, E. (eds.) CPM 2009. LNCS, vol. 5577, pp. 168–179. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02441-2_15
11. Clifford, P., Clifford, R., Iliopoulos, C.: Faster algorithms for δ, γ-matching and related problems. In: Apostolico, A., Crochemore, M., Park, K. (eds.) CPM 2005. LNCS, vol. 3537, pp. 68–78. Springer, Heidelberg (2005). https://doi.org/10.1007/11496656_7
12. Clifford, R.: Matrix multiplication and pattern matching under Hamming norm. http://www.cs.bris.ac.uk/Research/Algorithms/events/BAD09/BAD09/Talks/BAD09-Hammingnotes.pdf. Accessed Mar 2017
13. Clifford, R., Fontaine, A., Porat, E., Sach, B., Starikovskaya, T.: The k-mismatch problem revisited. In: SODA, pp. 2039–2052 (2016). https://doi.org/10.1137/1.9781611974331.ch142
14. Clifford, R., Kociumaka, T., Porat, E.: The streaming k-mismatch problem. In: SODA, pp. 1106–1125 (2019). https://doi.org/10.1137/1.9781611975482.68

15. Clifford, R., Starikovskaya, T.: Approximate hamming distance in a stream. In: ICALP, pp. 20:1–20:14 (2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.20
16. Fischer, M.J., Paterson, M.S.: String-matching and other products. Technical report (1974)
17. Gawrychowski, P., Uznański, P.: Towards unified approximate pattern matching for hamming and L_1 distance. In: ICALP, pp. 62:1–62:13 (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.62
18. Indyk, P.: Stable distributions, pseudorandom generators, embeddings, and data stream computation. J. ACM **53**(3), 307–323 (2006). https://doi.org/10.1145/1147954.1147955
19. Jayram, T.S., Kumar, R., Sivakumar, D.: The one-way communication complexity of hamming distance. Theory Comput. **4**(1), 129–135 (2008). https://doi.org/10.4086/toc.2008.v004a006
20. Karatsuba, A.: Multiplication of multidigit numbers on automata. Soviet physics doklady **7**, 595–596 (1963)
21. Karloff, H.J.: Fast algorithms for approximately counting mismatches. Inf. Process. Lett. **48**(2), 53–60 (1993). https://doi.org/10.1016/0020-0190(93)90177-B
22. Kopelowitz, T., Porat, E.: Breaking the variance: approximating the hamming distance in $1/\epsilon$ time per alignment. In: FOCS, pp. 601–613 (2015). https://doi.org/10.1109/FOCS.2015.43
23. Kopelowitz, T., Porat, E.: A simple algorithm for approximating the text-to-pattern hamming distance. In: SOSA@SODA, pp. 10:1–10:5 (2018). https://doi.org/10.4230/OASIcs.SOSA.2018.10
24. Kosaraju, S.R.: Efficient string matching (1987). Manuscript
25. Labib, K., Uznański, P., Wolleb-Graf, D.: Hamming distance completeness. In: CPM, pp. 14:1–14:17 (2019). https://doi.org/10.4230/LIPIcs.CPM.2019.14
26. Landau, G.M., Vishkin, U.: Efficient string matching with k mismatches. Theor. Comput. Sci. **43**, 239–249 (1986). https://doi.org/10.1016/0304-3975(86)90178-7
27. Lipsky, O., Porat, E.: Approximate matching in the L_∞ metric. Inf. Process. Lett. **105**(4), 138–140 (2008). https://doi.org/10.1016/j.ipl.2007.08.012
28. Lipsky, O., Porat, E.: L_1 pattern matching lower bound. Inf. Process. Lett. **105**(4), 141–143 (2008). https://doi.org/10.1016/j.ipl.2007.08.011
29. Lipsky, O., Porat, E.: Approximate pattern matching with the L_1, L_2 and L_∞ metrics. Algorithmica **60**(2), 335–348 (2011). https://doi.org/10.1007/s00453-009-9345-9
30. Nolan, J.: Stable Distributions: Models for Heavy-Tailed Data. Birkhauser, New York (2003)
31. Porat, B., Porat, E.: Exact and approximate pattern matching in the streaming model. In: FOCS, pp. 315–323 (2009). https://doi.org/10.1109/FOCS.2009.11
32. Porat, E., Efremenko, K.: Approximating general metric distances between a pattern and a text. In: SODA, pp. 419–427 (2008). http://dl.acm.org/citation.cfm?id=1347082.1347128
33. Starikovskaya, T., Svagerka, M., Uznański, P.: L_p pattern matching in a stream. CoRR abs/1907.04405 (2019)
34. Studený, J., Uznański, P.: Approximating approximate pattern matching. In: CPM, vol. 128, pp. 15:1–15:13 (2019). https://doi.org/10.4230/LIPIcs.CPM.2019.15
35. Toom, A.: The complexity of a scheme of functional elements simulating the multiplication of integers. In: Doklady Akademii Nauk, vol. 150, pp. 496–498. Russian Academy of Sciences (1963)

36. Uznański, P.: Approximating text-to-pattern distance via dimensionality reduction. CoRR abs/2002.03459 (2020)
37. Vassilevska, V.: Efficient algorithms for path problems in weighted graphs. Ph.D. thesis, Carnegie Mellon University (2008)
38. Woodruff, D.P.: Optimal space lower bounds for all frequency moments. In: SODA, pp. 167–175 (2004). http://dl.acm.org/citation.cfm?id=982792.982817
39. Zhang, P., Atallah, M.J.: On approximate pattern matching with thresholds. Inf. Process. Lett. **123**, 21–26 (2017). https://doi.org/10.1016/j.ipl.2017.03.001