Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity

Alexandr Andoni[*] Princeton University/C.C.I.

Robert Krauthgamer[†] Weizmann Institute

Krzysztof Onak[‡] MIT

April 28, 2010

Abstract

We present a near-linear time algorithm that approximates the edit distance between two strings within a polylogarithmic factor. More precisely, for strings of length n and every fixed $\varepsilon > 0$, it can compute a $(\log n)^{O(1/\varepsilon)}$-approximation in $n^{1+\varepsilon}$ time. This result arises naturally in the study of a new *asymmetric query* model. In this model, the input consists of two strings x and y, and an algorithm can access y in an unrestricted manner, while being charged for querying every symbol of x. Our query lower bound for this model provides the first rigorous separation between edit distance and Ulam distance, which is edit distance on non-repetitive strings, i.e., permutations.

1 Introduction

Manipulation of strings has long been central to computer science, which is abundant with requirements to efficiently process texts and other sequences. One of the first approaches to *comparing* two strings (sequences) emerged to be *edit distance* (aka Levenshtein distance) [Lev65], defined as the minimum number of character insertions, deletions, and substitutions needed to transform one string into the other. This basic distance measure is widely used in a variety of areas such as computational biology, speech recognition, and information retrieval. Consequently improvements in edit distance algorithms have the potential of major impact. As a result, computational problems involving edit distance were studied extensively, see [Nav01, Gus97], and references therein.

The most basic problem is that of computing the edit distance between two strings of length n over some alphabet. It can be solved in $O(n^2)$ time by a classical algorithm [WF74]; in fact this is a prototypical dynamic programming algorithm, see e.g. the textbook [CLRS01] and references therein. Despite significant research over nearly four decades, this running time has so far been improved only slightly to $O(n^2/\log^2 n)$ [MP80], which is the fastest algorithm known to date.[1]

Still, a near-quadratic runtime is often unacceptable in modern applications dealing with massive datasets such as the genomic data. Hence practitioners tend to rely on faster heuristics [Gus97,

[*]Supported in part by NSF CCF 0832797.

[†]Supported in part by The Israel Science Foundation (grant #452/08), and by a Minerva grant.

[‡]Supported in part by NSF grants 0732334 and 0728645.

[1]Their result applies to constant-size alphabets, and it was recently extended to arbitrarily large alphabets by Bille and Farach-Colton [BFC08] with an $O(\log\log n)^2$ factor loss in runtime.

Nav01]. This has motivated the quest for faster algorithms at the expense of approximation, see, e.g., [Ind01, Section 6] and [IM03, Section 8.3.2]. Indeed, the past decade has seen a serious effort in this direction.[2] One approach that is particularly relevant for us and may illustrate this general direction, is to design linear (or near-linear) time algorithms that approximate edit distance. A linear-time $\sqrt{n}$-approximation algorithm immediately follows from the exact algorithm of [LMS98], which runs in time $O(n + d^2)$, where d is the edit distance between the input strings. Subsequent research improved the approximation factor, first to $n^{3/7}$ [BJKK04], then to $n^{1/3+o(1)}$ [BES06], and finally to $2^{\tilde{O}(\sqrt{\log n})}$ [AO09] (building on [OR07]). Predating some of this work was the sublinear time algorithm of [BEK$^+$03] achieving n^ε approximation, but only when the edit distance d is rather large.

Better progress has been obtained on *variants* of edit distance, where one either restricts the input strings, or allows additional edit operations. One example for the first category is the case of edit distance on non-repetitive strings (e.g., permutations of $[n]$), termed *the Ulam distance* in the literature. The classical Patience Sorting algorithm computes the exact Ulam distance between two strings in $O(n \log n)$ time. An example in the second category is the case of two variants of edit distance with certain block operations allowed, which admit an $\tilde{O}(\log n)$-approximation in near-linear time [CPSV00, MS00, CM07, Cor03].

Yet, achieving a polylogarithmic approximation factor for the classical edit distance has eluded researchers for a long time now, not only in the context of linear-time algorithms, but also in the related tasks, such as nearest neighbor search, ℓ_1-embedding, or sketching. A *sublogarithmic* approximation has been ruled out for the latter two tasks [KN06, KR06, AK07], thus giving evidence that a sublogarithmic approximation for the distance computation might be much harder or even impossible to attain.

2 Results

Our first and main result is an algorithm that runs in near-linear time and approximates the classical edit distance within a *polylogarithmic factor*. Note that this is *exponentially better* than the previously known factor $2^{\tilde{O}(\sqrt{\log n})}$ (in comparable runtime), due to [OR07, AO09].

Theorem 2.1 (Main)**.** *For every fixed $\varepsilon > 0$, there is an algorithm that approximates the edit distance between two input strings $x, y \in \Sigma^n$ within a factor of $(\log n)^{O(1/\varepsilon)}$, and runs in $n^{1+\varepsilon}$ time.*

This development stems from a principled study of edit distance in a computational model, that we call the *asymmetric query* model, and which we shall define shortly. Specifically, we design a query-efficient procedure in the said model, and then show how this procedure yields a near-linear time algorithm. We also provide for this model a query complexity lower bound, which matches or nearly-matches the performance of our procedure. Altogether, we provide near-tight upper and lower bounds for this new query complexity model.

A conceptual contribution of our query complexity lower bound is that it is the first one to expose hardness stemming from "repetitive substrings", which means that many small substrings of a string may be approximately equal. Empirically, it is well-recognized that such repetitiveness

[2]We shall not attempt to present a complete list of results for restricted settings (e.g., average-case/smoothed analysis, weakly-repetitive strings, and bounded distance-regime), for variants of the distance function (e.g., allowing more edit operations), or for related computational problems (such as pattern matching, nearest neighbor, and sketching). See also the surveys of [Nav01] and [Sah08].

is a major obstacle for designing algorithms. All previous lower bounds (in any computational model) failed to exploit it, while in our proof the strings' repetitive structure is readily apparent. More formally, our lower bound provides the first rigorous separation of edit distance from Ulam distance (edit distance on permutations). Such a separation was not previously known in any studied model of computation, and in fact, all the lower bounds known for edit distance hold to (almost) the same degree for the Ulam distance. These models include: non-embeddability into normed spaces [KN06, KR06, AK07], lower bounds on sketching complexity [AK07, AJP10], and (symmetric) query complexity [BEK+03, AN10].

Asymmetric Query Complexity. Before stating the results formally, we define the problem and the model precisely. Consider two strings $x, y \in \Sigma^n$ for some alphabet Σ, and let $\mathrm{ed}(x, y)$ denote the edit distance between these two strings. The computational problem is the promise problem known as Distance Threshold Estimation Problem (DTEP) [SS02]: distinguish whether $\mathrm{ed}(x, y) > R$ or $\mathrm{ed}(x, y) \leq R/\alpha$, where $R > 0$ is a parameter (known to the algorithm) and $\alpha \geq 1$ is the *approximation factor*. We use DTEP_β to denote the case of $R = n/\beta$ where $\beta \geq 1$ may be a function of n.

In the *asymmetric query model*, the algorithm knows in advance (has unrestricted access to) one of the strings, say y, and has only *query access* to the other string, x. The *asymmetric query complexity* of an algorithm is the number of coordinates in x that the algorithm has to probe in order to solve DTEP with success probability at least 2/3.

We now give statements of our upper and lower bound results. Both exhibit a smooth *tradeoff* between approximation factor and query complexity. For simplicity, we state the bounds in two extreme regimes of approximation ($\alpha = \mathrm{polylog}(n)$ and $\alpha = \mathrm{poly}(n)$). Full statements are available in the full paper. Our upper bound is non-adaptive, whereas our lower bound holds for adaptive algorithms as well.

Theorem 2.2 (Query complexity upper bound). *For every $\beta = \beta(n) \geq 2$ and fixed $0 < \varepsilon < 1$ there is an algorithm that solves DTEP_β with approximation $\alpha = (\log n)^{O(1/\varepsilon)}$, and makes βn^ε asymmetric queries. This algorithm runs in time $O(n^{1+\varepsilon})$.*

For every $\beta = O(1)$ and fixed integer $t \geq 2$ there is an algorithm for DTEP_β achieving approximation $\alpha = O(n^{1/t})$, with $O(\log^{t-1} n)$ queries into x.

It is an easy observation that our general edit distance algorithm in Theorem 2.1 immediately follows from the above query complexity upper bound theorem, by running it for β equal to all powers of 2.

Theorem 2.3 (Query complexity lower bound). *For a sufficiently large constant $\beta > 1$, every algorithm that solves DTEP_β with approximation $\alpha = \alpha(n) > 2$ has asymmetric query complexity $2^{\Omega\left(\frac{\log n}{\log \alpha + \log\log n}\right)}$. Moreover, for every fixed non-integer $t > 1$, every algorithm that solves DTEP_β with approximation $\alpha = n^{1/t}$ has asymmetric query complexity $\Omega(\log^{\lfloor t \rfloor} n)$.*

Table 1 summarizes our results and previous bounds for DTEP_β under edit distance and Ulam distance. The more common query complexity model where the algorithm has query access to both strings is henceforth referred to as the symmetric model. We point out two implications of our bounds on the asymmetric query complexity:

- There is a strong separation between edit and Ulam distance. In the Ulam metric, a *constant* approximation is achievable with only $O(\log n)$ asymmetric queries (see [ACCL07], which builds on [EKK$^+$00]). For edit distance, we prove a much higher complexity, of $2^{\Omega(\log n/\log\log n)}$, even for a larger (polylogarithmic) approximation.

- Our query complexity upper and lower bound are nearly-matching, at least in some regime of parameters. At one extreme, approximation $O(n^{1/2})$ can be achieved with $O(\log n)$ queries, whereas approximation $n^{1/2-\varepsilon}$ already requires $\Omega(\log^2 n)$ queries. At the other extreme, approximation $\alpha = (\log n)^{1/\varepsilon}$ can be achieved using $n^{O(\varepsilon)}$ queries, and requires $n^{\Omega(\varepsilon/\log\log n)}$ queries.

Model	Metric	Approx.	Complexity	Remarks
Near-linear time	Edit	$(\log n)^{O(1/\varepsilon)}$	$n^{1+\varepsilon}$	Theorem 2.1
	Edit	$2^{\tilde{O}(\sqrt{\log n})}$	$n^{1+o(1)}$	[AO09]
Symmetric query complexity	Edit	n^{ε}	$\tilde{O}(n^{\max\{1-2\varepsilon,(1-\varepsilon)/2\}})$	[BEK$^+$03] (fixed β)
	Ulam	$O(1)$	$\tilde{O}(\beta+\sqrt{n})$	[AN10]
	Ulam+edit	$O(1)$	$\tilde{\Omega}(\beta+\sqrt{n})$	[AN10]
Asymmetric query complexity	Edit	$n^{1/t}$	$O(\log^{t-1} n)$	Theorem 2.2 (fixed $t \in \mathbb{N}, \beta > 1$)
	Edit	$n^{1/t}$	$\Omega(\log^{\lfloor t\rfloor} n)$	Theorem 2.3 (fixed $t \notin \mathbb{N}, \beta > 1$)
	Edit	$(\log n)^{1/\varepsilon}$	$\beta n^{O(\varepsilon)}$	Theorem 2.2
	Edit	$(\log n)^{1/\varepsilon}$	$n^{\Omega(\varepsilon/\log\log n)}$	Theorem 2.3 (fixed β)
	Ulam	$2+\varepsilon$	$O_{\varepsilon}(\beta\log\log\beta\cdot\log n)$	[ACCL07]

Table 1: Known results for DTEP$_\beta$ and arbitrary $0 < \varepsilon < 1$.

3 Connections to Previous Models

Symmetric Query Complexity. In a previously-studied model, the measure of complexity is how many positions the algorithm has to query in *both* strings (rather than only in one of the strings). Naturally, the query complexity in this model is at least as high as the query complexity in our model. This model has been proposed by [BEK$^+$03], and its main advantage is that it leads to *sublinear-time* algorithms for DTEP$_\beta$. The algorithm of [BEK$^+$03] makes $\tilde{O}(n^{1-2\varepsilon} + n^{(1-\varepsilon)/2})$ queries (and runs in the same time), and achieves n^{ε} approximation. However, it only works for the case $\beta = O(1)$.

In the symmetric query model, the known query lower bound is $\Omega(\sqrt{n/\alpha})$ for approximation α [BEK$^+$03, AN10], and comes from the same lower bound on Ulam distance. The lower bound essentially arises from the birthday paradox.

In terms of separating edit distance from the Ulam metric, this symmetric model can give at most a quadratic separation in the query complexity (since there exists a trivial algorithm with $2n$ queries). In contrast, in our asymmetric model, there is no "birthday paradox" lower bound, and the Ulam metric admits a constant approximation with $O(\log n)$ queries [EKK$^+$00, ACCL07]. Our lower bound for edit distance is exponentially bigger. This perhaps explains why it is much harder to design algorithms for edit distance than for Ulam distance.

Communication Complexity. In this setting, Alice and Bob each have a string, and they need to solve the DTEP$_\beta$ problem by way of exchanging messages. The measure of complexity is the number of bits exchanged in order to solve DTEP$_\beta$ with probability at least 2/3.

The best non-trivial upper bound known is $2^{\tilde{O}(\sqrt{\log n})}$ approximation with constant communication via [OR07, KOR00]. The only lower bound is that approximation α requires $\Omega(\frac{\log n \ / \ \log\log n}{\alpha})$ communication [AK07, AJP10].

The asymmetric model is "harder", in the sense that the query complexity is at least the communication complexity, up to a factor of $\log|\Sigma|$ in the complexity, since Alice and Bob can simulate the asymmetric query algorithm. In fact, our upper bound implies a communication protocol for the same DTEP_β problem with the same complexity, and where Alice sends just one message to Bob (i.e., it is a one-way communication protocol). This is the first communication protocol achieving polylogarithmic approximation for DTEP_β under edit distance.

4 Techniques

This section briefly highlights the main techniques and tools used in the course of proving our results.

Algorithm and Query Complexity Upper Bound. A high-level intuition for the near-linear time algorithm is as follows. The classical dynamic programming for edit distance runs in time that is the product of the lengths of the two strings. It seems plausible that, if we manage to "compress" one string to size n^ε, we may be able to compute the edit distance in time only $n^\varepsilon \cdot n$. Indeed, this is exactly what we accomplish. Specifically, our "compression" is achieved via a sampling procedure, which subsamples $\approx n^\varepsilon$ positions of x, and then computes $\mathrm{ed}(x, y)$ in time $n^{1+\varepsilon}$. Of course, the main challenge is, by far, how to subsample x so that the above is possible.

Our asymmetric query upper bound has two major components. The first component is a *characterization* of the edit distance by a different "distance", denoted $\mathcal{E}$, which approximates $\mathrm{ed}(x, y)$ well. The characterization is parametrized by an integer parameter $b \geq 2$ governing a tradeoff: small b leads to better approximation, whereas large b leads to faster algorithms. The second component is a *sampling algorithm* that approximates $\mathcal{E}$ for some settings of the parameter b, up to a constant factor, by making a small number of queries into x.

Our characterization is based on hierarchical decomposition of the edit distance computation, which is obtained by recursively partitioning the string x, each time into b blocks. We shall view this decomposition as a tree, whose arity is prescribed by the parameter b. Roughly speaking, the $\mathcal{E}$-distance at a node is then the sum, over all b children, of the minima of the $\mathcal{E}$-distances at those nodes over a certain range of displacements (possible "shifts" with respect to the other strings). At the leafs (corresponding to single characters of x), the $\mathcal{E}$-distance is simply the Hamming distance to corresponding positions in y.

We show that the characterization is an $O(\frac{b}{\log b}\log n)$ approximation to $\mathrm{ed}(x, y)$. Intuitively, the characterization manages to break-up the edit distance computation into *independent* distance computations on smaller substrings. The independence is crucial here as, one of the main computational challenges for the edit distance is the apparent need to find a *global* alignment between the two strings. We note that while the high-level approach, of recursively partitioning the strings, is somewhat similar to the previous approaches from [BEK+03, OR07, AO09], the technical development here is quite different. The previous results all relied on the following recurrence relation for the approximation factor α:

$$\alpha(n) = c \cdot \alpha(n/b) + \alpha(b),$$

for some $c \geq 2$. It is easy to see that one obtains $\alpha(n) \geq 2^{\Omega(\sqrt{\log n})}$ for any choice of $b \geq 2$. In contrast, our characterization is much more refined and achieves $c = 1$, which solves to $\alpha(n) = O(b \log_b n)$. Thus, our characterization may achieve even a *logarithmic* approximation.

The second component of our query algorithm is a careful sampling procedure that approximates the $\mathcal{E}$-distance up to a constant factor. The basic idea is to prune the above tree by subsampling at each node only a subset of its children. In particular, for a tree with arity $b = (\log n)^{1/\varepsilon}$, the hope is to subsample $(\log n)^{O(1)}$ children and use Chernoff-type bounds to argue that the subsample approximates well the $\mathcal{E}$-distance at that node. We note that $\Omega(\log n)$ samples of children is required due to the MIN operation taken at each node—we need the concentration bound, at each node, to hold with "high probability" to fight against a union bound. After such a pruning of the tree, we will be left with only $(\log n)^{O(\log_b n)} = n^{O(\varepsilon)}$ leafs, i.e., a few sampled positions of x.

However, this natural approach of subsampling $(\log n)^{O(1)}$ children at each node does not work when $\beta \gg 1$. Instead, we employ a *non-uniform subsampling technique*: for different nodes we subsample children at different, carefully-chosen rates. Our technique is somewhat reminiscent of the hierarchical decomposition and subsampling technique introduced by [IW05] in the context of sketching/streaming algorithms.

Query Complexity Lower Bound. The gist of our lower bound is designing two "hard distributions" $\mathcal{D}_0$ and $\mathcal{D}_1$, on strings in Σ^n, for which it is hard to distinguish, using only a few queries to x, whether $x \in \mathcal{D}_0$ or $x \in D_1$. At the same time, every two strings x, y in the support of the same $\mathcal{D}_i$ are at a small edit distance: $\mathrm{ed}(x, y) \leq n/(\alpha\beta)$; but for a mixed pair $x \in \mathcal{D}_0$ and $y \in D_1$, the distance is large: $\mathrm{ed}(x, y) > n/\beta$.

We start from the following core observation. Take two random strings $z_0, z_1 \in \{0,1\}^n$. Each $\mathcal{D}_i$, $i \in \{0,1\}$, is generated by applying a cyclic shift by a random $r \in [1, n/100]$ to the corresponding z_i. We show that in order to tell, for an input string, from which $\mathcal{D}_i$ it came from, one has to make at least $\Omega(\log n)$ queries. Intuitively, this comes from the fact that if the number q of queries is small, then the algorithm's view is close to the uniform distribution on $\{0,1\}^q$, no matter which positions are queried. Nevertheless, the edit distance between the two random strings is likely to be large, and a small shift will not change this significantly.

We then amplify the above query lower bound by applying the same idea recursively. In a string generated according to $\mathcal{D}_i$'s, we replace every symbol $a \in \{0,1\}$ by a random string selected independently from $\mathcal{D}_a$. This way we obtain two distributions on strings of length $n' = n^2$, that require $\Omega(\log^2 n) = \Omega(\log^2 n')$ queries to be told apart. We call the above operation of replacing symbols by strings that come from other distributions a *substitution product*. Strings created this way consist of n blocks of length n each. Intuitively, to tell the new distributions apart, one has to discover for at least $\Omega(\log n)$ blocks which distribution $\mathcal{D}_i$ they come from, to be able to tell which distribution $\mathcal{D}'_i$ the entire input comes from. By applying the recursive step multiple times, we achieve a $2^{\Omega(\frac{\log n}{\log\log n})}$ lower bound for a polylogarithmic approximation factor.

To formally prove this result, we develop several tools. In particular, we need tools for analyzing the behavior of edit distance under the product substitution. It turns out that to achieve a separation of edit distance between the hard distributions, we need a larger alphabet. After we are done with the recursion, we map the large alphabet to sufficiently long random binary strings, and thereby "extend" the lower bound to the binary alphabet.

We need tools also for analyzing indistinguishability of our distributions under a small number of queries. For this, we introduce a notion of similarity. This notion nicely composes with the substitution product, which amplifies the similarity. We also show that random acyclic shifts of random strings are likely to produce strings with high similarity. Finally, we show that any algorithm that tells apart distributions that meet our similarity notion must make many queries. We believe that these tools and ideas behind them may find applications in showing query lower bounds for other problems.

References

[ACCL07] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31:371–383, 2007. Previously appeared in RANDOM'04.

[AJP10] Alexandr Andoni, T.S. Jayram, and Mihai Pătraşcu. Lower bounds for edit distance and product metrics via Poincaré-type inequalities. *Accepted to ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, 2010.

[AK07] Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 724–734, 2007. Accepted to *SIAM Journal on Computing* (FOCS'07 special issue).

[AN10] Alexandr Andoni and Huy L. Nguyen. Near-tight bounds for testing Ulam distance. *Accepted to ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, 2010.

[AO09] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 199–204, 2009.

[BEK$^+$03] Tuğkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 316–324, 2003.

[BES06] Tuğkan Batu, Funda Ergün, and Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 792–801, 2006.

[BFC08] Philip Bille and Martin Farach-Colton. Fast and compact regular expression matching. *Theoretical Computer Science*, 409(28):486–496, 2008.

[BJKK04] Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2004.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 2nd edition, 2001.

[CM07] Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1), 2007. Special issue on SODA'02.

[Cor03] Graham Cormode. *Sequence Distance Embeddings.* Ph.D. Thesis, University of Warwick. 2003.

[CPSV00] Graham Cormode, Mike Paterson, Suleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 197–206, 2000.

[EKK$^+$00] Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Manesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000.

[Gus97] Dan Gusfield. *Algorithms on strings, trees, and sequences.* Cambridge University Press, Cambridge, 1997.

[IM03] Piotr Indyk and Jiří Matoušek. Low distortion embeddings of finite metric spaces. *CRC Handbook of Discrete and Computational Geometry*, 2003.

[Ind01] Piotr Indyk. Algorithmic aspects of geometric embeddings (tutorial). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 10–33, 2001.

[IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. *Proceedings of the Symposium on Theory of Computing (STOC)*, 2005.

[KN06] Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. Preliminary version appeared in FOCS'05.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. Preliminary version appeared in STOC'98.

[KR06] Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into L_1. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1010–1017, 2006.

[Lev65] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals (in russian). *Doklady Akademii Nauk SSSR*, 4(163):845–848, 1965. Appeared in English as: V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8), 707–710, 1966.

[LMS98] Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. Incremental string comparison. *SIAM J. Comput.*, 27(2):557–582, 1998.

[MP80] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980.

[MS00] S. Muthukrishnan and Cenk Sahinalp. Approximate nearest neighbors and sequence comparison with block operations. *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 416–424, 2000.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[OR07] Rafail Ostrovsky and Yuval Rabani. Low distortion embedding for edit distance. *J. ACM*, 54(5), 2007. Preliminary version appeared in STOC'05.

[Sah08] Süleyman Cenk Sahinalp. Edit distance under block operations. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

[SS02] Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 360–369, 2002.

[WF74] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168 – 173, 1974.