Edit Distance in Near Linear Time: $O(1)$ factor

Alex Andoni
Negev Shekel Nosatzki

(Columbia University)

Edit distance (Levenstein)

- Strings $x, y \in \Sigma^n$
- $ed_n(x, y) = \text{minimum number of insertions/deletions/substitutions to transform } x \text{ into } y$

$ed_n(\text{banana}, \text{ananas}) = 2$

Applications:
- bioinformatics
- natural language processing
Crucially: A classic dynamic programming

- Computing $ed_n(x, y)$:
  - $O(n^2)$ time [Wagner-Fischer’74]

\[
D(i, j) = \begin{cases} 
  ed(x[1:i], y[1:j]), & \text{if } x[i] = y[j] \\
  \min \left\{ D(i - 1, j) + 1, D(i, j - 1) + 1 \right\}, & \text{otherwise}
\end{cases}
\]
Faster Algorithms?

- **Computing** $ed_n(x, y)$:
  - $O(n^2)$ time [Wagner-Fischer’74]
  - $O(n^2 / \log^2 n)$ [MP’80]
  - Better in special cases (small $ed$, average case, smoothed, etc): [U83, LV85, M86, GG88, GP89, UW90, CL90, CH98, LMS98, U85, CL92, N99, CPS V00, MS00, CM02, BCF08, AK08, K’19…]
  - FGC: $n^{2-o(1)}$ likely best possible!
    - assuming Strong Exponential Time Hypothesis [BI’15, AHWW’16,…]

- **Approximation in near-linear time?**
  - $\log^{1/\epsilon} n$ factor in $n^{1+O(\epsilon)}$ time [BEKMRRS’03, BJKK’04, BES’06, AO’09, AKO’10]
  - $O(1)$ factor in $O(n^{1.781})$ quantum time [BEGHS’18]
  - $O(1)$ factor in $O(n^{1.618})$ time [CDGKS’18]
  - $O_\epsilon(1)$ factor & $\pm n^{1-f(\epsilon)}$ additive in $O(n^{1+\epsilon})$ time [KS’20, BR’20]
Main result

Can compute $ed(x, y)$ with $O_\epsilon(1)$ approx. in $n^{1+\epsilon}$ time

- Approach setup:
  - $ed_n(x, y) \iff$ an optimal alignment $\pi: [n] \to [n] \cup \{\bot\}$
  - $X_i, Y_j$: substrings starting at $i/j$ of length $w$ (think $w = n^{1-\delta}$)
  - Then $\sum_i \frac{ed_w(X_i, Y_{\pi(i)})}{w} \approx ed_n(x, y)$

**Goal:** find near-optimal matching $\pi$ between $X_i$’s and $Y_j$’s, using calls to $ed_w(X_i, Y_j)$ (possibly recursive)
Past approaches

Goal: find near-optimal matching $\pi$ between $X_i$’s and $Y_j$’s, using calls to $ed_w(X_i, Y_j)$

Why should help? [BEGHS’18, CDGKS’18]

- Naive compute-all: $n^2$ calls to $ed_w$ => time $n^2w^2$
  - Finding actual $\pi$: $(n/w)^{O(1)}$ time (~standard DP)
- Idea 0: enough to consider $i$ be multiple of $w$
  - Issue: $j = \pi(i)$ may not be $w$-multiple
  - Can round $j$ to $\delta w$, at the cost of additive $\delta n$ error
  - Reduces to $\approx (n/w)^2$ calls => time $\approx n^2$

$X_i$: interval of length $w$

Goal: find near-optimal matching $\pi$ between $X_i$’s and $Y_j$’s, using calls to $ed_w(X_i, Y_j)$

Why should help? [BEGHS’18, CDGKS’18]

- Naive compute-all: $n^2$ calls to $ed_w$ => time $n^2w^2$
  - Finding actual $\pi$: $(n/w)^{O(1)}$ time (~standard DP)
- Idea 0: enough to consider $i$ be multiple of $w$
  - Issue: $j = \pi(i)$ may not be $w$-multiple
  - Can round $j$ to $\delta w$, at the cost of additive $\delta n$ error
  - Reduces to $\approx (n/w)^2$ calls => time $\approx n^2$
Reducing # of calls to $ed_w(X_i, Y_j)$

**Goal:** find near-optimal matching $\pi$ between $X_i$’s and $Y_j$’s, using calls to $ed_w(X_i, Y_j)$

- **Idea 1:** use triangle inequality to deduce $ed_w(X_i, Y_j)$
  - If $X_i$ is “close” to $X_{i_1}, \ldots X_{i_m}$ and $Y_j$ “close” to $Y_{j_1}, \ldots Y_{j_m}$ => so are all of them, up to factor 2
  - Reduces # of $ed_w$ computations from $m^2$ to $\sim 2m$ (if ideal)!

- **Idea 2:** for $\pi(iw) = j$, most likely $\pi((i + 1)w) \approx j + w$

  + Idea 1,2 [CDGKS’18]: $(n/w)^{1.5}$ computations of $ed_w$!
  - Total time: $(n/w)^{1.5} \cdot w^2 + (n/w)^{O(1)}$

  + [KS’20, BR’20]: $(n/w)^{1+\epsilon}$ computations of $ed_w$
  - Extra $n^{1-f(\epsilon)}$ error term
  - E.g., allows to ignore a $n^{-f(\epsilon)}$ fraction of matches $X_i, Y_{\pi(i)}$

Or $\sim w^{1.5}$ if recursing on $ed_w$
Our high-level approach

- For each $w = 1, n^\varepsilon, n^{2\varepsilon}, ... n$,
- build a **distance oracle** $D_w$ for the metric $(\mathcal{I}_w, ed_w(\cdot;\cdot))$ where $\mathcal{I}_w = \text{all } 2n$ substrings of length $w$

---

**Oracle $D_w$:** for $I, J \in \mathcal{I}_w$

- $ed_w(I, J) \leq D_w(I, J)$
- $D_w(I, J) \preceq ed_w(I, J)$ where it “matters”
- $D_w(I, J)$ call takes $O^*(1)$ time

---

**New goal:** given $D_w/n^\varepsilon$, compute $D_w$, in $n^{1+O(\varepsilon)}$ time
2 components:

1. Alignment algo: oracle \( al_w(I, J) \)
2. Matching algo: building \( D_w \) from \( al_w \)

Distance \( G_w \) (shortest path): for \( I, J \in \mathcal{I}_w \)
- \( ed_w(I, J) \leq G_w(I, J) \)
- \( G_w(I, J) \simeq al_w(I, J) \) where it "matters"

\( ed_w(I, J) \leq al_w(I, J) \simeq ed_w(I, J) \) \( al_w(I, J) \) uses \( O^*(1) \) time & \( D_w/n^\epsilon \) calls

Any distance oracle OK, as long as metric output

E.g., [Thorup-Zwick’05] not metric output
But [Matousek’96]: embed into \( \ell^d_\infty \) where \( d = n^\epsilon = O^*(1) \) for approximation \( O(1/\epsilon) \)
Matching Algorithm

- Enough to build graph $G$ for **one scale** $c$:
  - Edge $(I, J)$ implies $al(I, J) \leq O_\epsilon(c)$
  - For any alignment $\pi$, for $i \in [n]$:  
    - If $al(X_i, Y_{\pi(i)}) \leq c$, there is a 2-hop path in $G$
    - Can miss $G$ if there $\leq$ number of pairs where $al(X_i, Y_{\pi(i)}) > c$
  - $O^*$ time and calls to $al$

**Perfect Neighborhood Assumption:**
- Either $al(I, J) \leq c$
- Or $al(I, J) \gg c$

I.e., $\mathcal{X}_W$ composed of equivalence classes

BIG simplification…

Instead of triangle inequality
Main loop

1. Iteratively partition \( \mathcal{S}_W \) into finer parts
   - In step \( t = 1 \ldots 1/\epsilon \), produce \( \Pi_t \)
   - \( \approx \lambda^t \) parts of size \( \approx n/\lambda^t \), for \( \lambda = n^\epsilon \)

2. Construction in step \( t \)
   - Sample \( \lambda^t \) anchors \( \in \mathcal{S}_W \) (each will produce a part in \( \Pi_t \))
   - For each anchor \( A \), compare to all in \( \Pi_{t-1}(A) \) using \( al \) oracle
   - Obtain set \( E(A) \) : all “equivalent” substrings (at distance \( \leq c \))
   - Each such \( I \in E \) is given credit \( \phi_A(I) = \frac{n/\lambda^t}{|E(A)|} \)

Fix part \( P \in \Pi_{t-1} \):
- Size \( \lambda \cdot n/\lambda^t \)
- About \( \lambda \) anchors inside
- Each should capture \( n/\lambda^t \)
Partition via proximity

- **Proximal extension** of $I \in E(A)$:
  - Distribute $\phi_A(I)$ to “nearby” $J$’s
  - $R$ intervals $J \in P_{t-1}(I)$ to left/right
  - Defines $\psi_A(J)$

- **New partition** $\Pi_t$ of $\mathcal{W}$:
  - Consider vectors $\psi(J) \in \mathbb{R}_+^{\lambda_t}$
  - Partition using (weighted) minhash $h: 2^{[\lambda_t]} \rightarrow [\lambda_t]$:
    - $J$ assigned to part $h(\psi(J))$
    - $\Pr[X_i$ and $Y_{\pi(i)}$ separated] $\approx ||\psi(X_i) - \psi(Y_{\pi(i)})||_1 \approx “local\ error”$

Repeat for $R = n^{\varepsilon l}$, for $l = 1$ ...

Each level $l$ “takes care” of intervals $I$ of density $E(I) \approx \Theta^* \left( \frac{n/\lambda^l}{n^{\varepsilon l}} \right)$

Use *thresholded* $\psi_A(J)$: zeroed-out if too small (to ensure no big parts)

Remove partitioned intervals from subsequent levels
A sample of the rest

- **Beyond “Perfect Neighborhood Assumption”:**
  - **Challenge:** can’t use usual ideas to reduce to PNA
    - E.g., if choose a random cut-off point $c$:
      - constant probability to separate $X_i$ from $Y_{\pi(i)}$ => like $ed \approx n$
    - Or FRT-like metric decomposition: $\Pr$ pair together $\approx n^{-\epsilon}$ not enough
    - Need a “for all” guarantee instead of “for each”
  1. **Smooth out everything:** “matching quantities” => up to $n^{O(\epsilon)}$
    - Eg, use *fractional* partitions (colorings): interval (logically) split b/w “parts”
    - New challenges to keep palettes sufficiently sparse
  2. Replace Jaccard (w-minhash) with “distortion resilient $\ell_1$”:
    - $dd_F(p, q) = \sum_i p_i \cdot \mathbb{I}[p_i > F \cdot q_i]$ for $F = n^{O(\epsilon)}$

- **Alignment Algorithm $al_w$:**
  - **Challenge 1:** $D_{w/n^\epsilon}$ arbitrary metric
  - **Challenge 2:** output of $al_w$ needs to be a metric

Perfect Neighborhood Assumption:
- Either $al(I,J) \leq c$
- Or $al(I,J) \gg c$
Finale

Can compute \( ed(x, y) \) with \( O_\epsilon(1) \) approx. in \( n^{1+\epsilon} \) time

- Approximation \( \sim \) doubly-exponential in \( 1/\epsilon \)

Open questions:

- \( \text{poly}(1/\epsilon) \) approximation?
  - Natural because using “dimension reduction” methods for metrics, where standard to have \( 2/\epsilon \) approx. vs \( n^\epsilon \) dimension

- Best runtime for \( 3 + \epsilon \) approximation?
  - E.g., \( \approx n^{1.5} \) natural: bottleneck is dynamic programming on substrings
  - Current best: \( \approx n^{1.6} \) \([\text{A’18, RSSS’19, GRS’20}]\)

- \( < 3 \) approximation (beyond triangle ineq)? \([\text{RSSS’19}]\)

- Many other edit distance problems:
  - Text indexing \([\text{CDK’19, A’18}]\), embedding/cutting modulus/NNS