Streaming algorithms for embedding and computing edit distance in the low distance regime

Diptarka Chakraborty
joint work with Elazar Goldenberg and Michal Koucký

15 January, 2016
Outline

1. Introduction
2. Result on Embedding
3. Applications
   - Document Exchange Problem
   - Other Applications
4. Embedding Algorithm
   - Description
   - Result
   - Proof Sketch
5. Computing Edit Distance
   - Comparison with Previous Work
   - Finding Alignment with Small Cost
   - Proof Sketch
1 Introduction

2 Result on Embedding

3 Applications
   • Document Exchange Problem
   • Other Applications

4 Embedding Algorithm
   • Description
   • Result
   • Proof Sketch

5 Computing Edit Distance
   • Comparison with Previous Work
   • Finding Alignment with Small Cost
   • Proof Sketch
Suppose we are given two strings \( x, y \in \{0, 1\}^* \)

- **Edit distance**, denoted by \( \Delta_e(x, y) \), is defined as the minimum number of insertion, deletion and bit flip operations needed for converting from \( x \) to \( y \).
Problem Definition

Suppose we are given two strings $x, y \in \{0, 1\}^*$

- **Edit distance**, denoted by $\Delta_e(x, y)$, is defined as the minimum number of insertion, deletion, and bit flip operations needed for converting from $x$ to $y$.

- **Hamming distance**, denoted by $\Delta_H(x, y)$, is defined as the minimum number of bit flip operations needed.
Problem Definition

Suppose we are given two strings $x, y \in \{0, 1\}^*$

- **Edit distance**, denoted by $\Delta_e(x, y)$, is defined as the minimum number of insertion, deletion and bit flip operations needed for converting from $x$ to $y$.

- **Hamming distance**, denoted by $\Delta_H(x, y)$, is defined as the minimum number of bit flip operations needed.

The problem is to find a *randomized embedding* from edit metric to Hamming metric with small *distortion factor*. 
Problem Definition

Define a map $f : \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$ s.t. $\forall x, y \in \{0, 1\}^n$,

$$\beta \cdot \Delta_e(x, y) \leq \Delta_H(f(x), f(y)) \leq \alpha \cdot \Delta_e(x, y)$$

where distortion factor is $\phi_d = \alpha / \beta$. 
Problem Definition

- Define a map $f : \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$ s.t. $\forall x, y \in \{0, 1\}^n$,
  $$\beta \cdot \Delta_e(x, y) \leq \Delta_H(f(x), f(y)) \leq \alpha \cdot \Delta_e(x, y)$$

  where distortion factor is $\phi_d = \alpha / \beta$.

- Now instead consider randomized map, i.e., $f : \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}^{l(n)}$ s.t. above holds w.h.p.
Problem Definition

- Define a map $f : \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$ s.t. $\forall x, y \in \{0, 1\}^n$, 

  \[
  \beta \cdot \Delta_e(x, y) \leq \Delta_H(f(x), f(y)) \leq \alpha \cdot \Delta_e(x, y)
  \]

  where distortion factor is $\phi_d = \alpha / \beta$.

- Now instead consider randomized map, i.e., 

  $f : \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}^{l(n)}$ s.t. above holds w.h.p.

**Remark:** Previous best known bound by Jowhari ’12: 

$\phi_d \leq O(\log n \log^* n)$
From computational perspective, problems on Hamming distance are somehow easier than that on edit distance.

Embedding provides us power to use results from the world of Hamming metric.
Motivation

- From computational perspective, problems on Hamming distance are somehow easier than that on edit distance
- Embedding provides us power to use results from the world of Hamming metric
- Applications include document exchange problem under edit metric, designing sketching protocol for gap-edit distance, approximately nearest neighbor search
1 Introduction

2 Result on Embedding

3 Applications
   • Document Exchange Problem
   • Other Applications

4 Embedding Algorithm
   • Description
   • Result
   • Proof Sketch

5 Computing Edit Distance
   • Comparison with Previous Work
   • Finding Alignment with Small Cost
   • Proof Sketch
Result on Embedding

There exists a mapping \( f : \{0, 1\}^n \times \{0, 1\}^{6n} \rightarrow \{0, 1\}^{3n} \) which satisfies the following conditions:

1. For every \( x, y \), \( \Delta_e(x, y) / 2 \leq \Delta_H(f(x, r), f(y, r)) \) with probability at least \( 1 - \exp(-\Omega(n)) \).
2. For every \( x, y \), \( \Delta_H(f(x, r), f(y, r)) \leq O(\Delta_e(x, y)) \) with probability at least \( 2/3 \).
3. Given \( f(x, r) \) and \( r \), it is possible to decode back \( x \) with probability \( 1 - \exp(-\Omega(n)) \). Moreover, both the mapping \( f \) and its decoding (given \( f(x, r) \) and \( r \)) take linear time and can be performed in a streaming fashion.
There exists a mapping \( f : \{0, 1\}^n \times \{0, 1\}^{6n} \rightarrow \{0, 1\}^{3n} \) which satisfies the following conditions:

1. For every \( x, y \), \( \Delta_e(x, y)/2 \leq \Delta_H(f(x, r), f(y, r)) \) with probability at least \( 1 - \exp(-\Omega(n)) \).
There exists a mapping $f : \{0, 1\}^n \times \{0, 1\}^{6n} \to \{0, 1\}^{3n}$ which satisfies the following conditions:

1. For every $x, y$, $\Delta_e(x, y)/2 \leq \Delta_H(f(x, r), f(y, r))$ with probability at least $1 - \exp(-\Omega(n))$.

2. For every $x, y$, $\Delta_H(f(x, r), f(y, r)) \leq O(\Delta_e(x, y))^2$ with probability at least $2/3$. 
Result on Embedding

There exists a mapping $f : \{0, 1\}^n \times \{0, 1\}^{6n} \rightarrow \{0, 1\}^{3n}$ which satisfies the following conditions:

1. For every $x, y$, $\Delta_e(x, y)/2 \leq \Delta_H(f(x, r), f(y, r))$ with probability at least $1 - \exp(-\Omega(n))$.

2. For every $x, y$, $\Delta_H(f(x, r), f(y, r)) \leq O(\Delta_e(x, y))^2$ with probability at least $2/3$.

3. Given $f(x, r)$ and $r$, it is possible to decode back $x$ with probability $1 - \exp(-\Omega(n))$. 
Result on Embedding

There exists a mapping $f : \{0, 1\}^n \times \{0, 1\}^{6n} \to \{0, 1\}^{3n}$ which satisfies the following conditions:

1. For every $x, y$, $\Delta_e(x, y)/2 \leq \Delta_H(f(x, r), f(y, r))$ with probability at least $1 - \exp(-\Omega(n))$.

2. For every $x, y$, $\Delta_H(f(x, r), f(y, r)) \leq O(\Delta_e(x, y))^2$ with probability at least $2/3$.

3. Given $f(x, r)$ and $r$, it is possible to decode back $x$ with probability $1 - \exp(-\Omega(n))$.

Moreover, both the mapping $f$ and its decoding (given $f(x, r)$ and $r$) take linear time and can be performed in a streaming fashion.
1. Introduction

2. Result on Embedding

3. Applications
   - Document Exchange Problem
   - Other Applications

4. Embedding Algorithm
   - Description
   - Result
   - Proof Sketch

5. Computing Edit Distance
   - Comparison with Previous Work
   - Finding Alignment with Small Cost
   - Proof Sketch

Diptarka Chakraborty, diptarka@cse.iitk.ac.in

Streaming algorithms for embedding and computing edit distance
Applications

- Computing edit distance (nontrivial)
- Document exchange problem under edit metric
- Designing sketching protocol for gap-edit distance
- Approximately nearest neighbor search
Document Exchange Problem

- Alice and Bob hold two strings $x$ and $y$ respectively.
- Bob’s task is to:
  - decide whether $\Delta_e(x, y) > k$
  - otherwise report $x$ correctly.
Use the following protocol (shared randomness) to solve:

- Alice and Bob compute $f(x, r)$ and $f(y, r)$ respectively (linear time)
- Use protocol for Hamming metric by Porat and Lipsky ’07 (uses $O(k^2 \log n + n \log n)$ time and $O(k^2 \log n)$ bits to be transmitted)
- Bob will learn $f(x, r)$ and then decode (linear time)
- Use algorithm by Landau et al. to decide whether $\Delta_e(x, y) \leq k$ ($O(n + k^2)$ time)
Document Exchange Problem

Use the following protocol (shared randomness) to solve:

- Alice and Bob compute $f(x, r)$ and $f(y, r)$ respectively (linear time)
- Use protocol for Hamming metric by Porat and Lipsky ’07 (uses $O(k^2 \log n + n \log n)$ time and $O(k^2 \log n)$ bits to be transmitted)
- Bob will learn $f(x, r)$ and then decode (linear time)
- Use algorithm by Landau et al. to decide whether $\Delta_e(x, y) \leq k$ ($O(n + k^2)$ time)

**Remark:** Previous best known bound by Jowhari ’12: $O(n \log n + k^2 \log^2 n)$ on time and $O(k \log^2 n \log^* n)$ on number of bits to be transmitted
Similarly,

- solves $k$ vs. $ck^2$ gap-edit distance, for some $c > 0$ using constant size sketches
- return a point within the distance $O(k)$ times that of the closest one
1 Introduction

2 Result on Embedding

3 Applications
   - Document Exchange Problem
   - Other Applications

4 Embedding Algorithm
   - Description
   - Result
   - Proof Sketch

5 Computing Edit Distance
   - Comparison with Previous Work
   - Finding Alignment with Small Cost
   - Proof Sketch
The basic scheme is as follows:

- Pick a sequence of random functions $h_1, \ldots, h_{3n} : \{0, 1\} \rightarrow \{0, 1\}$
- Maintain a pointer $i$ for current position on input and initially set to 1
- In time $t \leq 3n$, append output by $x_i$ and increment $i$ by $h_t(x_i)$
- If $i$ exceeds $n$, append zeros
Example

Suppose $h_1(0) = 0$, $h_1(1) = 1$; $h_2(0) = 0$, $h_2(1) = 0$; $h_3(0) = 1$, $h_3(1) = 0$; \ldots
Consider $x = 10110 \cdots$
Example

Suppose $h_1(0) = 0, \ h_1(1) = 1; \ h_2(0) = 0, \ h_2(1) = 0; \ h_3(0) = 1, \ h_3(1) = 0; \cdots$

Consider $x = 10110 \cdots$

$output = 1, \ i = 1, \ i = i + h_1(1) = 2$
Example

Suppose $h_1(0) = 0, h_1(1) = 1$; $h_2(0) = 0, h_2(1) = 0$; $h_3(0) = 1$, $h_3(1) = 0$; \ldots
Consider $x = 10110 \cdots$
\begin{align*}
\text{output} &= 1, \ i = 1, \ i = i + h_1(1) = 2 \\
\text{output} &= 10, \ i = 2, \ i = i + h_2(0) = 2
\end{align*}
Example

Suppose $h_1(0) = 0$, $h_1(1) = 1$; $h_2(0) = 0$, $h_2(1) = 0$; $h_3(0) = 1$, $h_3(1) = 0$; \ldots
Consider $x = 10110 \cdots$
\begin{align*}
\text{output} &= 1, \quad i = 1, \quad i = i + h_1(1) = 2 \\
\text{output} &= 10, \quad i = 2, \quad i = i + h_2(0) = 2 \\
\text{output} &= 100, \quad i = 2, \quad i = i + h_3(0) = 3
\end{align*}
Example

Suppose \( h_1(0) = 0, \ h_1(1) = 1; \ h_2(0) = 0, \ h_2(1) = 0; \ h_3(0) = 1, \ h_3(1) = 0; \ \cdots \)

Consider \( x = 10110 \cdots \)

\( \text{output} = 1, \ i = 1, \ i = i + h_1(1) = 2 \)
\( \text{output} = 10, \ i = 2, \ i = i + h_2(0) = 2 \)
\( \text{output} = 100, \ i = 2, \ i = i + h_3(0) = 3 \)
\( \text{output} = 1001, \ i = 3, \ \cdots \)
Recall the Result

There exists a mapping $f : \{0, 1\}^n \times \{0, 1\}^{6n} \rightarrow \{0, 1\}^{3n}$ which satisfies the following conditions:

1. For every $x, y$, $\Delta_e(x, y)/2 \leq \Delta_H(f(x, r), f(y, r))$ with probability at least $1 - \exp(-\Omega(n))$.

2. For every $x, y$, $\Delta_H(f(x, r), f(y, r)) \leq O(\Delta_e(x, y))^2$ with probability at least $2/3$.

3. Given $f(x, r)$ and $r$, it is possible to decode back $x$ with probability $1 - \exp(-\Omega(n))$.

Moreover, both the mapping $f$ and its decoding (given $f(x, r)$ and $r$) take linear time and can be performed in a streaming fashion.
Proof Sketch

- Use Chernoff bound to show that $i$ will exceed $n$ within time $3n$ except with probability $2^{-\Omega(n)}$
Proof Sketch

- Use Chernoff bound to show that $i$ will exceed $n$ within time $3n$ except with probability $2^{-\Omega(n)}$
- If exceed, then use the similar algorithm to decode
Proof Sketch

- Use Chernoff bound to show that $i$ will exceed $n$ within time $3n$ except with probability $2^{-\Omega(n)}$
- If exceed, then use the similar algorithm to decode
- Lower bound follows from decoding algorithm
Proof Sketch

- Use Chernoff bound to show that $i$ will exceed $n$ within time $3n$ except with probability $2^{-\Omega(n)}$.
- If exceed, then use the similar algorithm to decode.
- Lower bound follows from decoding algorithm.
- To prove upper bound, reduce the problem to a well-known problem on random walk.
Idea behind the Upper Bound

Suppose edit distance is 2 (one insertion and one deletion at the end)
Idea behind the Upper Bound

- Suppose edit distance is 2 (one insertion and one deletion at the end)
- Consider the first index $i$ where $x_i \neq y_i$
Idea behind the Upper Bound

- Suppose edit distance is 2 (one *insertion* and one deletion at the end)
- Consider the first index $i$ where $x_i \neq y_i$
- With probability $1/2$, $i$ is incremented in $x$ and same for $y$
Idea behind the Upper Bound

- Suppose edit distance is 2 (one insertion and one deletion at the end)
- Consider the first index $i$ where $x_i \neq y_i$
- With probability $1/2$, $i$ is incremented in $x$ and same for $y$
- With probability $1/4$ they got synced, with probability $1/2$ defer by 1 and with remaining probability defer by 2
Idea behind the Upper Bound

- Suppose edit distance is 2 (one \textit{insertion} and one deletion at the end)
- Consider the first index $i$ where $x_i \neq y_i$
- With probability 1/2, $i$ is incremented in $x$ and same for $y$
- With probability 1/4 they got synced, with probability 1/2 defer by 1 and with remaining probability defer by 2
- Same as random walk on integer line starting from origin and probability of sync within $l$ steps is same as probability of visiting position 1 for the first time within $l$ steps
Idea behind the Upper Bound

- Suppose edit distance is 2 (one **insertion** and one deletion at the end)
- Consider the first index $i$ where $x_i \neq y_i$
- With probability $1/2$, $i$ is incremented in $x$ and same for $y$
- With probability $1/4$ they got synced, with probability $1/2$ defer by 1 and with remaining probability defer by 2
- Same as random walk on integer line starting from origin and probability of sync within $l$ steps is same as probability of visiting position 1 for the first time within $l$ steps
- For constant probability, we need $l$ to be constant
1 Introduction

2 Result on Embedding

3 Applications
   - Document Exchange Problem
   - Other Applications

4 Embedding Algorithm
   - Description
   - Result
   - Proof Sketch

5 Computing Edit Distance
   - Comparison with Previous Work
   - Finding Alignment with Small Cost
   - Proof Sketch
## Comparison with Previous Work

<table>
<thead>
<tr>
<th>Authors</th>
<th>Time</th>
<th>Space</th>
<th>Approx. Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>WF74</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
<td>Exact</td>
</tr>
<tr>
<td>LMS98</td>
<td>$O(n + k^2)$</td>
<td>$O(n)$</td>
<td>Exact</td>
</tr>
<tr>
<td>PP08</td>
<td>$O(n + k^2)$</td>
<td>$O(n)$</td>
<td>Exact</td>
</tr>
<tr>
<td>LMS98</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>AKO10</td>
<td>$n^{1+\epsilon}$ (randomized)</td>
<td>$O(n)$</td>
<td>$(\log n)^{O(1/\epsilon)}$</td>
</tr>
<tr>
<td>Saha14</td>
<td>$O(n)$ (randomized)</td>
<td>$O(\log n)$</td>
<td>$O(k)$</td>
</tr>
<tr>
<td><strong>This paper</strong></td>
<td>$O(n + k^6)$ (randomized with promise)</td>
<td>$O(k^6)$</td>
<td>Exact</td>
</tr>
</tbody>
</table>
Finding Alignment with small cost

- Maintain two pointers $i_x, i_y$ for $x$ and $y$ respectively
- If $x_{i_x} = y_{i_y}$, set $a(i_x) = i_y$
- Else with probability $1/2$, set $a(i_x) = D$ and increment $i_x$
- With remaining probability increment only $i_y$
- Stop if both $i_x$ and $i_y$ reach $n + 1$
- If $i_y = n + 1$ and $i_x < n$, set $a(i_x), \cdots, a(n) = D$
Proof Sketch

- Compute an alignment of cost $ck^2$ (linear time)
Proof Sketch

- Compute an alignment of cost $ck^2$ (linear time)
- Compute kernels of size $O(k^6)$ for each of the inputs (linear time)
Proof Sketch

- Compute an alignment of cost $ck^2$ (linear time)
- Compute kernels of size $O(k^6)$ for each of the inputs (linear time)
- Run known algorithm, say PP08, on those kernels ($O(k^6 + k^2)$)
Proof Sketch

- Compute an alignment of cost \(ck^2\) (linear time)
- Compute kernels of size \(O(k^6)\) for each of the inputs (linear time)
- Run known algorithm, say PP08, on those kernels \(O(k^6 + k^2)\)
- Probability of success is at least 2/3
Kernelization

Lemma (Deflation)

Let \( x, y \in \{0, 1\}^n \). Let \( x = uwv \) and \( y = u'wv' \) for some strings \( u, w, v, u', v' \). Let \( K \) and \( k \) be integers such that \( \Delta_e(x, y) \leq k \) and \( ||u|| - ||u'|| \leq K \). Let \( \ell \) be the minimal period of \( w \) and \( p \in \{0, 1\}^\ell, r > 0 \) be such that \( w = p^r \). Let \( t = 2K + 3k + (\ell + 2) \cdot (k + 1) \). If \( |w| \geq t \) then for all \( r' \) such that \( r' \geq t/\ell \), \( \Delta_e(x, y) = \Delta_e(up^{r'} v, u'p^{r'} v') \).
Kernelization

Lemma (Shrinkage)

Let $x, y \in \{0, 1\}^n$. Let $x = uwv$ and $y = u'wv'$ for some strings $u, w, v, u', v'$. Let $K, k$ and $t$ be integers such that $\Delta_e(x, y) \leq k$, $||u| - |u'|| \leq K$, and assume $w$ is $(t, K + k)$-periodic free. Let $s = K + 2k + (k + 1) \cdot (t + 1)$. For any $s' \geq s$, if $|w| \geq 2s'$ and $w' = w_1, \ldots, s'w|w|+1-s', \ldots, |w|$ then $\Delta_e(x, y) = \Delta_e(uw'v, u'w'v')$. 
Algorithm to Compute Kernel

Input: $x, y \in \{0, 1\}^n$ such that $\Delta_e(x, y) \leq k$ and an alignment $a$ of cost at most $ck^2$.

Output: $x', y' \in \{0, 1\}^n$ such that $\Delta_e(x', y') = \Delta_e(x, y)$.

Decompose $x = u_0 w_1 u_1 \cdots w_\ell u_\ell$ and $y = v_0 w_1 v_1 \cdots w_\ell v_\ell$ where $\ell \leq ck^2 + 1$, each $w_i$ is a maximal preserved block of $x$ under $a$.

Deflate each $w_i$ so that no $c_1k^3$ block is periodic with period $c_2k^2$ (Use Knuth-Morris-Pratt algorithm).

Shrink each $w_i$ to keep only the first and last $c_3k^4$ length portion (Use some cyclic buffer).
Algorithm to Compute Kernel

Input: $x, y \in \{0, 1\}^n$ such that $\Delta_e(x, y) \leq k$ and an alignment $a$ of cost at most $ck^2$.
Output: $x', y' \in \{0, 1\}^{O(k^6)}$ such that $\Delta_e(x', y') = \Delta_e(x, y)$. 
Algorithm to Compute Kernel

Input: $x, y \in \{0, 1\}^n$ such that $\Delta_e(x, y) \leq k$ and an alignment $a$ of cost at most $ck^2$.
Output: $x', y' \in \{0, 1\}^{O(k^6)}$ such that $\Delta_e(x', y') = \Delta_e(x, y)$.

- Decompose $x = u_0 w_1 u_1 \ldots w_\ell u_\ell$ and $y = v_0 w_1 v_1 \ldots w_\ell v_\ell$
Algorithm to Compute Kernel

Input: \( x, y \in \{0,1\}^n \) such that \( \Delta_e(x, y) \leq k \) and an alignment \( a \) of cost at most \( ck^2 \).

Output: \( x', y' \in \{0,1\}^{O(k^6)} \) such that \( \Delta_e(x', y') = \Delta_e(x, y) \).

- Decompose \( x = u_0w_1u_1 \cdots w_\ell u_\ell \) and \( y = v_0w_1v_1 \cdots w_\ell v_\ell \) where \( \ell \leq ck^2 + 1 \), each \( w_i \) is a maximal preserved block of \( x \) under \( a \).
Algorithm to Compute Kernel

Input: $x, y \in \{0, 1\}^n$ such that $\Delta_e(x, y) \leq k$ and an alignment $a$ of cost at most $ck^2$.

Output: $x', y' \in \{0, 1\}^{O(k^6)}$ such that $\Delta_e(x', y') = \Delta_e(x, y)$.

- Decompose $x = u_0 w_1 u_1 \cdots w_\ell u_\ell$ and $y = v_0 w_1 v_1 \cdots w_\ell v_\ell$ where $\ell \leq ck^2 + 1$, each $w_i$ is a maximal preserved block of $x$ under $a$.

- Deflate each $w_i$ so that no $c_1 k^3$ block is periodic with period $c_2 k^2$ (Use Knuth-Morris-Pratt algorithm).
Algorithm to Compute Kernel

Input: $x, y \in \{0, 1\}^n$ such that $\Delta_e(x, y) \leq k$ and an alignment $a$ of cost at most $ck^2$.
Output: $x', y' \in \{0, 1\}^{O(k^6)}$ such that $\Delta_e(x', y') = \Delta_e(x, y)$.

- Decompose $x = u_0 w_1 u_1 \cdots w_\ell u_\ell$ and $y = v_0 w_1 v_1 \cdots w_\ell v_\ell$ where $\ell \leq ck^2 + 1$, each $w_i$ is a maximal preserved block of $x$ under $a$.
- Deflate each $w_i$ so that no $c_1 k^3$ block is periodic with period $c_2 k^2$ (Use Knuth-Morris-Pratt algorithm).
- Shrink each $w_i$ to keep only first and last $c_3 k^4$ length portion (Use some cyclic buffer).
Boosting Probability

- To boost probability, re-run the embedding and kernelization if alignment is too costly
Boosting Probability

- To boost probability, re-run the embedding and kernelization if alignment is too costly
- Use pre-computed kernel to avoid re-reading input
Boosting Probability

- To boost probability, re-run the embedding and kernelization if alignment is too costly
- Use pre-computed kernel to avoid re-reading input
- To achieve probability $1 - 1/n$, we need to consider $O(\log n)$ many alignments and thus incurring $O(n + k^6 \log n)$ time and $O(k^6)$ space
Boosting Probability

- To boost probability, re-run the embedding and kernelization if alignment is too costly
- Use pre-computed kernel to avoid re-reading input
- To achieve probability $1 - 1/n$, we need to consider $O(\log n)$ many alignments and thus incurring $O(n + k^6 \log n)$ time and $O(k^6)$ space
- Everything discussed so far can be implemented in one-pass streaming model
Boosting Probability

- To boost probability, re-run the embedding and kernelization if alignment is too costly
- Use pre-computed kernel to avoid re-reading input
- To achieve probability $1 - 1/n$, we need to consider $O(\log n)$ many alignments and thus incurring $O(n + k^6 \log n)$ time and $O(k^6)$ space
- Everything discussed so far can be implemented in one-pass streaming model
- Non-promise version requires $O(\log \log k)$ passes
THANK YOU!!!