Edit Distance: Sketching, Streaming and Document Exchange

Djamal Belazzougui  
CERIST, Algeria

Qin Zhang  
IU Bloomington

FOCS 2016  
Oct. 9, 2016
Definition: Given two strings $s, t \in \Sigma^n$:

$$ed(s, t) = \text{minimum number of character operations (insertion/deletion/substitution) that transform } s \text{ to } t.$$
Definition: Given two strings $s, t \in \Sigma^n$:

$ed(s, t) =$ minimum number of character operations (insertion/deletion/substitution) that transform $s$ to $t$.

$ed(\text{banana, ananas}) = 2$
**Definition:** Given two strings $s, t \in \Sigma^n$:

$$ed(s, t) = \text{minimum number of character operations (insertion/deletion/substitution) that transform } s \text{ to } t.$$ 

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

**Applications:** numerous. E.g.,

bioinformatics (measuring similarity between DNA seq.)
Definition: Given two strings \( s, t \in \Sigma^n \):

\[ ed(s, t) = \text{minimum number of character operations (insertion/deletion/substitution) that transform } s \text{ to } t. \]

\[ ed( \text{banana}, \text{ananas} ) = 2 \]

Applications: numerous. E.g.,

bioinformatics (measuring similarity between DNA seq.)

automatic spelling correction
The threshold version of ED: Given two strings \( s, t \in \{0, 1\}^n \) and a threshold \( K \), output all the edits if \( ed(s, t) \leq K \), output “Error” otherwise.
The threshold version of ED: Given two strings \( s, t \in \{0, 1\}^n \) and a threshold \( K \), output all the edits if \( ed(s, t) \leq K \), output “Error” otherwise.

Models/Problems:

\[ s \xrightarrow{sk(s)} t \]

document exchange
App: remote file sync;
file transmission through
a noisy channel
The threshold version of ED: Given two strings $s, t \in \{0, 1\}^n$ and a threshold $K$, output all the edits if $ed(s, t) \leq K$, output “Error” otherwise.

Models/Problems:

- **Document exchange**
  App: remote file sync; file transmission through a noisy channel

- **sketching**
  App: distributed similarity join
The threshold version of ED: Given two strings $s, t \in \{0, 1\}^n$ and a threshold $K$, output all the edits if $ed(s, t) \leq K$, output “Error” otherwise.

Models/Problems:

**document exchange**
App: remote file sync; file transmission through a noisy channel

**streaming**

**sketching**
App: distributed similarity join
### Previous and our results

<table>
<thead>
<tr>
<th>problem</th>
<th>comm. / size / space (bits)</th>
<th>running time</th>
<th>rand. or det.</th>
<th>ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>document-exchange</td>
<td>$O(K \log n)$</td>
<td>$n^{O(K)}$</td>
<td>D</td>
<td>[23]</td>
</tr>
<tr>
<td></td>
<td>$O(K \log(n/K) \log n)$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>[18]</td>
</tr>
<tr>
<td></td>
<td>$O(K \log^2 n \log^* n)$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>[19]</td>
</tr>
<tr>
<td></td>
<td>$O(K^2 + K \log^2 n)$</td>
<td>$\tilde{O}(n)$</td>
<td>D</td>
<td>[5]</td>
</tr>
<tr>
<td></td>
<td>$O(K^2 \log n)$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>$O(K(\log^2 K + \log n))$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>new</td>
</tr>
<tr>
<td>sketching</td>
<td>$O(K^8 \log^5 n)$</td>
<td>$\tilde{O}(K^2 n)$ (enc.), poly($K \log n$) (dec.)</td>
<td>R</td>
<td>new</td>
</tr>
<tr>
<td>streaming</td>
<td>$O(K^8 \log^5 n)$</td>
<td>$\tilde{O}(K^2 n)$</td>
<td>R</td>
<td>new</td>
</tr>
<tr>
<td>simultaneous-streaming</td>
<td>$O(K^6 \log n)$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>$O(K \log n)$</td>
<td>$O(n)$</td>
<td>D</td>
<td>new</td>
</tr>
</tbody>
</table>

$K$: distance threshold; $n$: input size. For simplicity, assuming $K < n^{0.1}$

- **Information theoretic optimal communication** for $K \leq 2^{\sqrt{\log n}}$ under almost linear encoding/decoding time for doc-exchange.

- **First** sketching/streaming algorithm with $\text{poly}(K, \log n)$ size/space.

Note: $\Omega(n)$ LB for linear sketches. (Andoni, Goldberger, McGregor, Porat. STOC'13)
Previous and our results

<table>
<thead>
<tr>
<th>problem</th>
<th>comm. / size / space (bits)</th>
<th>running time</th>
<th>rand. or det.</th>
<th>ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>document-exchange</td>
<td>$O(K \log n)$</td>
<td>$n^{O(K)}$</td>
<td>D</td>
<td>[23]</td>
</tr>
<tr>
<td></td>
<td>$O(K \log(n/K) \log n)$</td>
<td>$O(n)$</td>
<td>R</td>
<td>[18]</td>
</tr>
<tr>
<td></td>
<td>$O(K \log^2 n \log^* n)$</td>
<td>$O(n)$</td>
<td>R</td>
<td>[19]</td>
</tr>
<tr>
<td></td>
<td>$O(K^2 + K \log^2 n)$</td>
<td>$O(n)$</td>
<td>D</td>
<td>[5]</td>
</tr>
<tr>
<td></td>
<td>$O(K^2 \log n)$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>$O(K(\log^2 K + \log n))$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>new</td>
</tr>
<tr>
<td>sketching</td>
<td>$O(K^8 \log^5 n)$</td>
<td>$\tilde{O}(K^2 n)$ (enc.), poly($K \log n$) (dec.)</td>
<td>R</td>
<td>new</td>
</tr>
<tr>
<td>streaming</td>
<td>$O(K^8 \log^5 n)$</td>
<td>$\tilde{O}(K^2 n)$</td>
<td>R</td>
<td>new</td>
</tr>
<tr>
<td>simultaneous-streaming</td>
<td>$O(K^6 \log n)$</td>
<td>$\tilde{O}(n)$</td>
<td>R</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>$O(K \log n)$</td>
<td>$O(n)$</td>
<td>D</td>
<td>new</td>
</tr>
</tbody>
</table>

$K$: distance threshold; $n$: input size. For simplicity, assuming $K < n^{0.1}$

- **Information theoretic optimal communication** for $K \leq 2^{\sqrt{\log n}}$ under almost linear encoding/decoding time for doc-exchange.
- **First sketching/streaming algorithm** with poly($K$, log $n$) size/space.

Note: $\Omega(n)$ LB for linear sketches. (Andoni, Goldberger, McGregor, Porat. STOC'13)
Main Tool:
CGK Embedding
The CGK embedding

\( f : s \in \{0, 1\}^n \rightarrow s' \in \{0, 1\}^{3n} \).

Two counters \( i \) and \( j \) both initialized to 1. For \( j = 1, 2, \ldots \) steps:
1. \( s'[j] \leftarrow s[i] \).
2. Flip a coin; if head, then \( i \leftarrow i + 1 \). Stop when \( i = n + 1 \).
3. \( j \leftarrow j + 1 \).

(Chakraborty, Goldenberg, Koucky, STOC’16
Similar idea by Saha, FOCS’14)
Our main tool – CGK embedding

- The CGK embedding

  \[ f : s \in \{0, 1\}^n \rightarrow s' \in \{0, 1\}^{3n}. \]

  Two counters \( i \) and \( j \) both initialized to 1. For \( j = 1, 2, \ldots \) steps:
  1. \( s'[j] \leftarrow s[i] \).
  2. Flip a coin; if head, then \( i \leftarrow i + 1 \). Stop when \( i = n + 1 \).
  3. \( j \leftarrow j + 1 \).

- Property

  If \( ed(s, t) = k \), then \( k/2 \leq \text{ham}(f(s), f(t)) \leq O(k^2) \) w.pr. 0.99

(Chakraborty, Goldenberg, Koucky, STOC’16
Similar idea by Saha, FOCS’14)
CGK as a random walk

- CGK \rightarrow \text{ a random walk on two strings}

\[\begin{align*}
\text{s} & \quad 101\cdot\cdot \quad \text{CGK} \quad 10111\cdot\cdot \\
\text{t} & \quad 11\cdot\cdot\cdot \quad \text{CGK} \quad 11111\cdot\cdot\cdot \\
\end{align*}\]
CGK as a random walk

- CGK → a random walk on two strings

- The shift \((p - q)\) is a random walk on the line.
Document Exchange

App: remote file sync; file transmission through a noisy channel

Warning: I will cheat in multiple places
Main idea: If we can find $\leq K$ pairs of blocks in $s$ and $t$ each of size $K^{99}$, such that they contain all the edits, then IMS gives $O(K(\log^2 K))$. (recall IMS gives $O(K \log n \log(n/K))$)
Main idea: If we can find $\leq K$ pairs of blocks in $s$ and $t$ each of size $K^{99}$, such that they contain all the edits, then IMS gives $O(K(\log^2 K))$. (recall IMS gives $O(K \log n \log(n/K))$)

Question: if exist, how to identify these pairs?
Main idea: If we can find $\leq K$ pairs of blocks in $s$ and $t$ each of size $K^{99}$, such that they contain all the edits, then IMS gives $O(K(\log^2 K))$. (recall IMS gives $O(K \log n \log(n/K))$)

Question: if exist, how to identify these pairs?

CGK (edit-space $\rightarrow$ ham-space) + random partition to blocks + error-correcting code for Ham w.r.t. blocks + reverse mapping
Main idea: If we can find \( \leq K \) pairs of blocks in \( s \) and \( t \) each of size \( K^{99} \), such that they contain all the edits, then IMS gives \( O(K(\log^2 K)) \). (recall IMS gives \( O(K \log n \log(n/K)) \))

Question: if exist, how to identify these pairs?

CGK
(edit-space \( \rightarrow \) ham-space) + random partition to blocks + error-correcting code for Ham w.r.t. blocks + reverse mapping

Challenge: the \( O(K^2) \) errors after CGK embedding can possibly be distributed into \( O(K^2) \) pairs of blocks. This may introduce a factor of \( K^2 \) of communication in the error-correcting which is too much.
Main idea: If we can find \( \leq K \) pairs of blocks in \( s \) and \( t \) each of size \( K^{99} \), such that they contain all the edits, then IMS gives \( O(K(\log^2 K)) \). (recall IMS gives \( O(K \log n \log(n/K)) \))

Question: if exist, how to identify these pairs?

\begin{align*}
\text{CGK} & \quad (\text{edit-space} \rightarrow \text{ham-space}) + \text{random partition to blocks} + \\
& \quad \text{error-correcting code for Ham w.r.t. blocks} + \text{reverse mapping}
\end{align*}

Challenge: the \( O(K^2) \) errors after CGK embedding can possibly be distributed into \( O(K^2) \) pairs of blocks. This may introduce a factor of \( K^2 \) of communication in the error-correcting which is too much.

- Can reduce \( O(K^2) \) pairs to \( O(K) \), by removing long common periodic substrings.
- Not easy: everything has to be done using one-way comm.!
Technique overview: document exchange (cont.)

\[
s \quad \begin{array}{c|c|c|c|c|c} \hline 1 & 0 & 1 & \cdot & \cdot & \cdot \\ \hline \end{array} \\
\text{CGK} \\
\begin{array}{c|c|c|c|c|c} \hline 1 & 0 & 1 & 1 & 1 & \cdot \cdot \\ \hline \end{array}
\]

\[
t \quad \begin{array}{c|c|c|c|c|c} \hline 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ \hline \end{array} \\
\text{CGK} \\
\begin{array}{c|c|c|c|c|c} \hline 1 & 1 & 1 & 1 & 1 & \cdot \cdot \\ \hline \end{array}
\]
Call a walk step from state \((p, q)\) a progress step if \(s[p] \neq t[q]\) and one of these cases happens.
Technique overview: document exchange (cont.)

- Call a walk step from state \((p, q)\) a progress step if \(s[p] \neq t[q]\) and one of these cases happens:
- Call a seq. of walks from state \((p, q)\) where the next progress step happens, to the first state \((p', q')\) where
\[
ed(s[p'...n], t[q'...n]) = ed(s[p...n], t[q...n]) - 1
\] a progress phase.
Technique overview: document exchange (cont.)

Call a walk step from state \((p, q)\) a \textit{progress step} if \(s[p] \neq t[q]\)
and one of these cases happens

Call a seq. of walks from state \((p, q)\) where the next progress step
happens, to the first state \((p', q')\) where
\[
ed(s[p'...n], t[q'...n]) = ed(s[p...n], t[q...n]) - 1
\]
a progress phase

a progress phase \(\Leftrightarrow\) a pair of mismatching blocks

\(\leq K\) progress phases \(\Rightarrow\) \(\leq K\) pairs of mismatching blocks

\# random walk steps in a progress phase \(\Leftrightarrow\)
size of the mismatching block
Call a seq. of walks from state \((p, q)\) where a (the next) progress step happens, to the first state \((p', q')\) where
\[
ed(s[p'...n], t[q'...n]) = ed(s[p...n], t[q...n]) - 1\]
a progress phase

\(\leq K\) progress phases \(\Rightarrow\) \(\leq K\) pairs of mismatching blocks

\# random walk steps in a progress phase \(\iff\) size of the mismatching block

Whp, a progress phase “consumes” \(\leq K^{10}\) progress steps.
Technique overview: document exchange (cont.)

- Call a seq. of walks from state \((p, q)\) where a (the next) progress step happens, to the first state \((p', q')\) where
  \[ ed(s[p'...n], t[q'...n]) = ed(s[p...n], t[q...n]) - 1 \]
  a progress phase

- \(\leq K\) progress phases \(\Rightarrow\) \(\leq K\) pairs of mismatching blocks

- \(\#\) random walk steps in a progress phase \(\iff\) size of the mismatching block

- Whp, a progress phase “consumes” \(\leq K^{10}\) progress steps.

- Can show that after properly removing long common periods, we get a progress step in \(\leq K^{50}\) random walk steps
Technique overview: document exchange (cont.)

- Call a seq. of walks from state \((p, q)\) where a (the next) progress step happens, to the first state \((p', q')\) where 
  \[ ed(s[p'...n], t[q'...n]) = ed(s[p...n], t[q...n]) - 1 \] 
  a progress phase

- \(\leq K\) progress phases \(\Rightarrow\) \(\leq K\) pairs of mismatching blocks

- \# random walk steps in a progress phase \(\iff\) size of the mismatching block

- Whp, a progress phase “consumes” \(\leq K^{10}\) progress steps.

- Can show that after properly removing long common periods, we get a progress step in \(\leq K^{50}\) random walk steps

Recall our main idea: If we can find \(\leq K\) pairs of blocks in \(s\) and \(t\) each of size \(K^{99}\), such that they contain all the edits, then IMS gives \(O(K(\log^2 K))\). (Other steps cost \(O(K \log n)\))
Sketching

App: distributed similarity join
We can view an alignment $\mathcal{A}$ between $s$ and $t$ as a non-crossing bipartite matching

\[
\begin{array}{cccccccc}
  s & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
  t & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{array}
\]
We can view an alignment $\mathcal{A}$ between $s$ and $t$ as a non-crossing bipartite matching.

\[
\begin{array}{cccccccc}
\text{s} & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\text{t} & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{array}
\]

Can be compressed by writing down all singletons and starting/ending edges of each cluster, denoted by $sk(\mathcal{A})$. 
We can view an alignment $A$ between $s$ and $t$ as a non-crossing bipartite matching

$$
\begin{array}{cccccccccc}
  s & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
  \hline
  t & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{array}
$$

Can be compressed by writing down all singletons and starting/ending edges of each cluster, denoted by $sk(A)$.

Note: size of $sk(OPT)$ is only $O(K \log n)$.
Technique overview: sketching

- We can view an alignment $\mathcal{A}$ between $s$ and $t$ as a non-crossing bipartite matching:

  $\begin{align*}
  s & \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \\
  t & \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 
  \end{align*}$

  Can be compressed by writing down all singletons and starting/ending edges of each cluster, denoted by $sk(\mathcal{A})$.

  Note: size of $sk(OPT)$ is only $O(K \log n)$.

- Given alignments $\mathcal{A}_1, \ldots, \mathcal{A}_\rho$, letting $\mathcal{I} = \bigcap_{j \in [\rho]} \mathcal{A}_j$

  **Main idea:** if $\exists$ an optimal alignment that goes through all edges in $\mathcal{I}$, then we can obtain an optimal alignment using $sk(\mathcal{A}_1), \ldots, sk(\mathcal{A}_\rho)$.
CGK embedding naturally gives an alignment.

The random walk state sequence \(((p_1, q_1), (p_2, q_2), \ldots)\) contains an alignment \(A\), which can be constructed in a greedy way, and \(sk(A)\) has size \(\text{poly}(K, \log n)\).
CGK embedding naturally gives an alignment.

The random walk state sequence \(((p_1, q_1), (p_2, q_2), \ldots)\) contains an alignment \(\mathcal{A}\), which can be constructed in a greedy way, and \(sk(\mathcal{A})\) has size \(\text{poly}(K, \log n)\).

**Key lemma:** Can show if we take \(\rho = \text{poly}(K, \log n)\) random walks which give alignments \(\mathcal{A}_1, \ldots, \mathcal{A}_\rho\), then there is an optimal alignment contains \(\mathcal{I} = \bigcap_{j \in [\rho]} \mathcal{A}_j\)
CGK embedding naturally gives an alignment.

The random walk state sequence \(((p_1, q_1), (p_2, q_2), \ldots)\) contains an alignment \(\mathcal{A}\), which can be constructed in a greedy way, and \(sk(\mathcal{A})\) has size \(\text{poly}(K, \log n)\).

**Key lemma:** Can show if we take \(\rho = \text{poly}(K, \log n)\) random walks which give alignments \(\mathcal{A}_1, \ldots, \mathcal{A}_\rho\), then there is an optimal alignment contains \(\mathcal{I} = \bigcap_{j \in [\rho]} \mathcal{A}_j\).

\(sk(\mathcal{A}_i) \Leftrightarrow\) differences between \(s'\) and \(t'\) in the ham-space for which efficient sketching algo is known.
CGK embedding naturally gives an alignment.

The random walk state sequence \(((p_1, q_1), (p_2, q_2), \ldots)\) contains an alignment \(\mathcal{A}\), which can be constructed in a greedy way, and \(sk(\mathcal{A})\) has size \(\text{poly}(K, \log n)\).

**Key lemma**: Can show if we take \(\rho = \text{poly}(K, \log n)\) random walks which give alignments \(\mathcal{A}_1, \ldots, \mathcal{A}_\rho\), then there is an optimal alignment contains \(\mathcal{I} = \bigcap_{j \in [\rho]} \mathcal{A}_j\)

\(sk(\mathcal{A}_i) \Leftrightarrow\) differences between \(s'\) and \(t'\) in the ham-space for which efficient sketching algo is known.

Additional structures needed for the reverse mapping (ham-space \(\rightarrow\) edit-space) to find all the edits.
We have obtained

- Information theoretic optimal communication (for small $K$) under almost linear encoding/decoding time for document exchange.
- First sketching/streaming algorithm with $\text{poly}(K, \log n)$ size/space.
Conclusion and open problems

- We have obtained
  - Information theoretic optimal communication (for small $K$) under almost linear encoding/decoding time for document exchange.
  - First sketching/streaming algorithm with $\text{poly}(K, \log n)$ size/space.

- Open problems
  - For doc-exchange, can we further improve the communication to the information-theoretic optimal bound $O(K \log n)$ for all values $K$ and $n$ under (almost) linear running time?
Conclusion and open problems

- We have obtained
  - Information theoretic optimal communication (for small $K$) under almost linear encoding/decoding time for document exchange.
  - First sketching/streaming algorithm with $\text{poly}(K, \log n)$ size/space.

- Open problems
  - For doc-exchange, can we further improve the communication to the information-theoretic optimal bound $O(K \log n)$ for all values $K$ and $n$ under (almost) linear running time?
  - For sketching, what are the best dependencies on $K$ and $\log n$? Can we prove any non-trivial lower bounds? (Now $K^8 \log^5 n$. We believe with a more careful analysis on the same algo, can reduce to $K^4 \log^3 n$ or even $K^3 \log^2 n$)
Conclusion and open problems

- We have obtained
  - Information theoretic optimal communication (for small $K$) under almost linear encoding/decoding time for document exchange.
  - First sketching/streaming algorithm with poly($K$, $\log n$) size/space.

- Open problems
  - For doc-exchange, can we further improve the communication to the information-theoretic optimal bound $O(K \log n)$ for all values $K$ and $n$ under (almost) linear running time?
  - For sketching, what are the best dependencies on $K$ and $\log n$? Can we prove any non-trivial lower bounds? (Now $K^8 \log^5 n$. We believe with a more careful analysis on the same algo, can reduce to $K^4 \log^3 n$ or even $K^3 \log^2 n$)
  - Is it possible to derandomize our algorithm for doc-exchange to obtain a better error-correcting code for edit distance?
Thank you! Questions?
Key lemma: Can show if we take \( \rho = \operatorname{poly}(K, \log n) \) random walks which give alignments \( A_1, \ldots, A_\rho \), then there is an optimal alignment contains \( I = \bigcap_{j \in [\rho]} A_j \)

- Anchor. Given \( \rho \) random walks generated according to the CGK embedding, we say a pair \((u, v)\) is an anchor if \( s[u] = t[v] \), and all the \( \rho \) random walks pass \((u, v)\).

- Claim: W.pr. \( 1 - 1/n^2 \), there is an optimal alignment going through all anchors.

- Proof idea: We focus on a “greedy” optimal matching \( O \). Suppose on the contrary that \( O \) does not pass an anchor \((u, v)\), then we can find a matching \( M \) in the left neighborhood of \((u, v)\) which may “mislead” a random walk, that is, with a non-trivial probability the random walk will “follow” \( M \) and consequently miss \((u, v)\).