Why compute minimum edit distance?

Sometimes we want to know how “similar” two strings are.

- Could indicate morphological relationships:
  - walk - walks, sleep - slept

- Or possible spelling errors (and corrections):
  - definition - defintion, separate - separate

- Also used in other fields, e.g., bioinformatics (gene sequences):
  - ACCGTA - ACCGATA

MED is (one) way to measure similarity

- How many changes needed to go from string $s_1 \rightarrow s_2$?

  $\begin{align*}
  S & \quad T & A & L & L \\
  T & A & L & L & \text{deletion} \\
  T & A & B & L & \text{substitution} \\
  T & A & B & L & E & \text{insertion}
  \end{align*}$

- To solve the problem, we need to find the best alignment between the words.
  - Could be several equally good alignments.

Alignments and edit distance

These two problems reduce to one: find the optimal character alignment between two words (the one with the fewest character changes: the minimum edit distance or MED).

- Example: if all changes count equally, MED(stall, table) is 3:

  $\begin{align*}
  S & \quad T & A & L & L \\
  T & A & L & L & \text{deletion} \\
  T & A & B & L & \text{substitution} \\
  T & A & B & L & E & \text{insertion}
  \end{align*}$
Alignments and edit distance

These two problems reduce to one: find the optimal character alignment between two words (the one with the fewest character changes: the minimum edit distance or MED).

- Example: if all changes count equally, MED(stall, table) is 3:

<table>
<thead>
<tr>
<th>S</th>
<th>T</th>
<th>A</th>
<th>L</th>
<th>L</th>
<th>d</th>
<th>i</th>
<th>s</th>
<th>i</th>
<th>d</th>
<th>i</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>A</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>A</td>
<td>B</td>
<td>L</td>
<td>substitution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>A</td>
<td>B</td>
<td>L</td>
<td>E</td>
<td>insertion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Written as an alignment:

  S T A L L -
  d | | s | i
  - T A B L E

More alignments

- There may be multiple best alignments. In this case, two:

  S T A L L -
  d | | s | i
  - T A B L E

  S T A L L -
  s d | i d
  - T A B L E

- And lots of non-optimal alignments, such as:

  S T A L L -
  d d s s i
  - - T A B L E

How to find an optimal alignment

Brute force: Consider all possibilities, score each one, pick best.

How many possibilities must we consider?

- First character could align to any of:

  - - - - - T A B L E -

- Next character can align anywhere to its right

- And so on... the number of alignments grows exponentially with the length of the sequences.

Maybe not such a good method...

A better idea

Instead we will use a dynamic programming algorithm.

- Other DP (or memoization) algorithms we’ll see later: Viterbi, CKY.

- Used to solve problems where brute force ends up recomputing the same information many times.

- Instead, we

  - Compute the solution to each subproblem once,
  - Store (memoize) the solution, and
  - Build up solutions to larger computations by combining the pre-computed parts.

- Strings of length \( n \) and \( m \) require \( O(mn) \) time and \( O(mn) \) space.
**Intuition**

- Minimum distance $D(\text{stall, table})$ must be the minimum of:
  - $D(\text{stall, tabl}) + \text{cost(ins)}$
  - $D(\text{stal, table}) + \text{cost(del)}$
  - $D(\text{stal, tabl}) + \text{cost(sub)}$

- Similarly for the smaller subproblems

- So proceed as follows:
  - solve smallest subproblems first
  - store solutions in a table (chart)
  - use these to solve and store larger subproblems until we get the full solution

---

**A note about costs**

- Our first example had $\text{cost(ins)} = \text{cost(del)} = \text{cost(sub)} = 1$.

- But we can choose whatever costs we want. They can even depend on the particular characters involved.
  - Ex: choose $\text{cost(sub(c,c')}))$ to be $P(c'|c)$, the probability of someone accidentally typing $c'$ when they meant to type $c$.
  - Then we end up computing the most probable sequence of typos that would change one word to the other.

- In the following example, we’ll assume $\text{cost(ins)} = \text{cost(del)}= 1$ and $\text{cost(sub)} = 2$.

---

**Chart: starting point**

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Chart[$i, j$] stores two things:
  - $D(\text{stall}[0..i], \text{table}[0..j])$: the MED of substrings of length $i, j$
  - Backpointer(s) showing which sub-alignment(s) was/were extended to create this one.

---

**Filling first cell**

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Moving down in chart: means we had a deletion (of S).
- That is, we’ve aligned (S) with (-).
- Add cost of deletion (1) and backpointer.

---
Each move down first column means another deletion.

- $D(ST, -) = D(S, -) + \text{cost(del)}$

Moving right in chart (from [0,0]): means we had an insertion.

- That is, we’ve aligned (-) with (T).
- Add cost of insertion (1) and backpointer.

Moving down and right: either a substitution or identity.

- Here, a substitution: we aligned (S) to (T), so cost is 2.
- For identity (align letter to itself), cost is 0.
### Multiple paths

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>←1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>↑1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>↑2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>↑3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>↑4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>↑5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- However, we also need to consider other ways to get to this cell:
  - Move **down** from [0,1]: deletion of S, total cost is $D(-, T) + \text{cost(del)} = 2$.
  - Same cost, but add a new backpointer.

### Single best path

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>←1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>↑1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>↑2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>↑3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>↑4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>↑5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Now compute $D(ST, T)$. Take the min of three possibilities:
  - $D(ST, -) + \text{cost(ins)} = 2 + 1 = 3$.
  - $D(S, T) + \text{cost(del)} = 2 + 1 = 3$.
  - $D(S, -) + \text{cost(ident)} = 1 + 0 = 1$.

### Final completed chart

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>A</th>
<th>B</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>←1</td>
<td>←2</td>
<td>←3</td>
<td>←4</td>
<td>←5</td>
</tr>
<tr>
<td>S</td>
<td>↑1</td>
<td>←↖↑2</td>
<td>←↖↑3</td>
<td>←↖↑4</td>
<td>←↖↑5</td>
</tr>
<tr>
<td>T</td>
<td>↑2</td>
<td>←↖↑1</td>
<td>←↖↑2</td>
<td>←↖↑3</td>
<td>←↖↑4</td>
</tr>
<tr>
<td>A</td>
<td>↑3</td>
<td>↑2</td>
<td>↑1</td>
<td>←↖↑1</td>
<td>←↖↑2</td>
</tr>
<tr>
<td>L</td>
<td>↑4</td>
<td>↑3</td>
<td>↑2</td>
<td>←↖↑3</td>
<td>←↖↑2</td>
</tr>
</tbody>
</table>

- Follow the backpointers to find the best alignment(s). This path, for example, corresponds to: $S$ $T$ $A$ $- L$ $L$ $-$ $d$ $| i d i d$ $| i $ $- T$ $A$ $B$ $- L$ $E$
Exercises

• Choose a different path through the backpointers and reconstruct its alignment.

• How many different optimal alignments are there?

• Redo the chart with all costs = 1 (Levenshtein distance), or some other set of costs, or using a different word pair.