Minimum edit distance: worked example

Sharon Goldwater

15 September 2017

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 - seperate

- 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 $s_1 \rightarrow s_2$?

S	T	A	L	L	
	T	A	L	L	deletion
	T	A	B	L	substitution
	T	A	B	L E	insertion

- 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:

S	T	A	L	L	
	T	A	L	L	deletion
	T	A	B	L	substitution
	T	A	B	L E	insertion

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:

S	T	A	L	L		
	T	A	L	L		deletion
	T	A	B	L		substitution
	T	A	B	L	E	insertion

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

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

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

S	T	A	L	-	L	-
d	d	s	s	i	|	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(stall, table) must be the minimum of:
  - D(stall, tabl) + cost(ins)
  - D(stal, table) + cost(del)
  - D(stal, tabl) + 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 cost(ins) = cost(del) = cost(sub) = 1.
- But we can choose whatever costs we want. They can even depend on the particular characters involved.
  - Ex: choose 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 cost(ins) = cost(del)= 1 and cost(sub) = 2.

Chart: starting point

		T	A	B	L	E
	0					
S						
T						
A						
L						
L						

- 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

		T	A	B	L	E
	0					
S	↑1					
T						
A						
L						
L						

- 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.

Rest of first column

		T	A	B	L	E
	0					
S	↑1					
T	↑2					
A						
L						
L						

- Each move down first column means another deletion.
  - D(ST, -) = D(S, -) + cost(del)

Rest of first column

		T	A	B	L	E
	0					
S	↑1					
T	↑2					
A	↑3					
L	↑4					
L	↑5					

- Each move down first column means another deletion.
  - D(ST, -) = D(S, -) + cost(del)
  - D(STA, -) = D(ST, -) + cost(del)
  - etc

Start of second column: insertion

		T	A	B	L	E
	0	←1				
S	↑1					
T	↑2					
A	↑3					
L	↑4					
L	↑5					

- 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.

Substitution

		T	A	B	L	E
	0	←1				
S	↑1	↖2				
T	↑2					
A	↑3					
L	↑4					
L	↑5					

- 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

		T	A	B	L	E
	0	←1				
S	↑1	↖↑2				
T	↑2					
A	↑3					
L	↑4					
L	↑5					

- 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) + cost(del) = 2.
  - Same cost, but add a new backpointer.

Multiple paths

		T	A	B	L	E
	0	←1				
S	↑1	←↖↑2				
T	↑2					
A	↑3					
L	↑4					
L	↑5					

- However, we also need to consider other ways to get to this cell:
  - Move **right** from [1,0]: insertion of T, total cost is D(S, -) + cost(ins) = 2.
  - Same cost, but add a new backpointer.

Single best path

		T	A	B	L	E
	0	←1				
S	↑1	←↖↑2				
T	↑2	↖1				
A	↑3					
L	↑4					
L	↑5					

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

Final completed chart

		T	A	B	L	E
	0	←1	←2	←3	←4	←5
S	↑1	←↖↑2	←↖↑3	←↖↑4	←↖↑5	←↖↑6
T	↑2	↖1	←2	←3	←4	←5
A	↑3	↑2	↖1	←2	←3	←4
L	↑4	↑3	↑2	←↖↑3	↖2	←3
L	↑5	↑4	↑3	←↖↑4	↖↑3	←↖↑4

- Follow the backpointers to find the best alignment(s). This path, for example, corresponds to:

S	T	A	-	L	L	-
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.