# Minimum Edit Distance: Second Look

Jean Mark Gawron

Linguistics 522
San Diego State University

2018 FEB

# Outline

1. Introduction

# leda deal

| | # | *l* | *e* | *d* | *a* |
|---|---|---|---|---|---|
| *l* | 4 | 3 | | | |
| *a* | 3 | 4 | ?? | | |
| *e* | 2 | 3 | ? | | |
| *d* | 1 | 2 | 3 | | |
| # | 0 | 1 | 2 | 3 | 4 |
| | # | *l* | *e* | *d* | *a* |

**target**

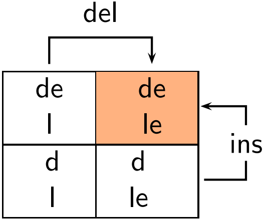
$$
\begin{aligned}
?? \quad D(3,2) &= min(D(3,1)+1\\
&\qquad\quad D(2,1)+2,\\
&\qquad\quad D(2,2)+1)\\
? \quad D(2,2) &= min(D(2,1)+1,\\
&\qquad\quad D(1,1)+0,\\
&\qquad\quad D(1,2)+1)
\end{aligned}
$$

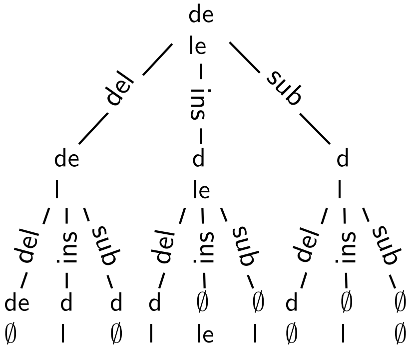
the alignment cell (2,2) ? represents de<br>le

the alignment cell (3,2) ?? represents dea<br>le

# Sources of an alignment

| | | | | | |
|---|---|---|---|---|---|
| source | d | e | 0 | a | l |
| target | l | e | d | a | 0 |
| | sub | sub | ins | sub | del |




$$\begin{array}{c} de \\ l \end{array} \rightarrow \begin{array}{c} de \\ le \end{array} \begin{pmatrix} de & 0 \\ l & e \end{pmatrix}$$

target **e** aligned with 0: deletion

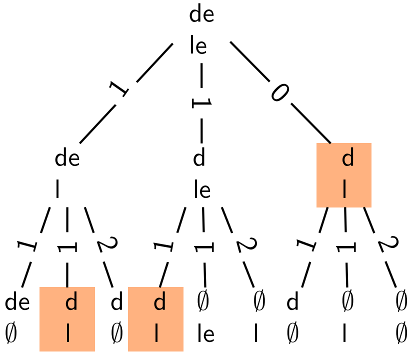
# Edit problems and alignments: The computational graph

a. Each node is an edit-distance problem (a **state** in the computational graph).

b. **Paths** from a node to the leaves are alignments (solutions).

c. Same state crops up on different paths.

d. Naive traversal from root to leaves solves same problem multiple times. (Order $3^{m+n}$)



# The big picture

1. Computational graph, drawn as tree, is actually a DAG (Directed Acyclic Graph), because the same problem recurs in multiple places.
2. So the Minimal Edit Distance algorithm is solving the problem of finding the shortest distance through a weighted graph. (Order $m \times n$)
3. A special case of what we will call the **Viterbi** algorithm later in the course.