Foundations of Natural Language Processing
Lecture 8b
Spelling Correction and Edit Distance

Alex Lascarides

School of informatics

Recap: A simple noise model for spelling correction

- Where y is the intended word and x is the (perhaps misspelled) word, we want:

$$\operatorname*{argmax}_y P(y|x) \quad = \quad \operatorname*{argmax}_y P(x|y)P(y)$$

- Possible y restricted to 'one character difference' with x.

$$\begin{aligned} P(y): &\quad \text{Language model} \\ P(x|y) = &\quad \textstyle\prod_{i=1}^{n} P(x_i|y_i) \end{aligned}$$

- Learn $P(x|y)$ from a corpus of character alignments.

$$\begin{array}{lcccccccccccccccc} \text{actual:} & \mathrm{n} & \mathrm{o} & \text{-} & & \mathrm{m} & \mathrm{u} & \mathrm{u} & \mathrm{c} & \mathrm{h} & & \mathrm{e} & \mathrm{f} & \mathrm{f} & \mathrm{e} & \mathrm{r} & \mathrm{t} \\ & | & | & | & | & | & | & | & | & | & | & | & | & | & | & | & | \\ \text{intended:} & \mathrm{n} & \mathrm{o} & \mathrm{t} & & \mathrm{m} & \text{-} & \mathrm{u} & \mathrm{c} & \mathrm{h} & & \mathrm{e} & \mathrm{f} & \mathrm{f} & \mathrm{o} & \mathrm{r} & \mathrm{t} \end{array}$$

Problems

1. Independence assumption is unrealistic.

2. Assumption restricting possible intended words is unrealistic.

3. We may not have a corpus of alignments!

Now: Approach that solves problems 1 and 2: edit distance
(Solve problem 3 later. . .)

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	$\vert$	$\vert$	s	$\vert$	i
-	T	A	B	L	E

S	T	A	-	L	L
d	$\vert$	$\vert$	i	$\vert$	s
-	T	A	B	L	E

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

S	T	A	-	L	-	L
s	d	$\vert$	i	$\vert$	i	d
T	-	A	B	L	E	-

S	T	A	L	-	L	-
d	d	s	s	i	$\vert$	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: 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.
  - For example: choose cost(sub(c,c')) to be $P(c'|c)$ from our spelling correction noise model!
  - Then we end up computing the most probable way to 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):** which sub-alignment(s) used to create this one.

Deletion:	Move down	Cost =1
Insertion:	Move right	Cost=1
Substitution:	Move down and right	Cost=2 (or 0 if the same)

Sum costs as we expand out from cell (0,0) to populate the entire matrix

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	$\uparrow 1$					
T	$\uparrow 2$					
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	$\uparrow 1$					
T	$\uparrow 2$					
A	$\uparrow 3$					
L	$\uparrow 4$					
L	$\uparrow 5$					

- 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	$\leftarrow 1$				
S	$\uparrow 1$	$\nwarrow 2$				
T	$\uparrow 2$					
A	$\uparrow 3$					
L	$\uparrow 4$					
L	$\uparrow 5$					

- 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	$\leftarrow$1				
S	$\uparrow$1	$\nwarrow\uparrow$2				
T	$\uparrow$2					
A	$\uparrow$3					
L	$\uparrow$4					
L	$\uparrow$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	$\leftarrow$1				
S	$\uparrow$1	$\leftarrow\nwarrow\uparrow$2				
T	$\uparrow$2					
A	$\uparrow$3					
L	$\uparrow$4					
L	$\uparrow$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	$\leftarrow 1$				
S	$\uparrow 1$	$\leftarrow\nwarrow\uparrow 2$				
T	$\uparrow 2$	$\nwarrow 1$				
A	$\uparrow 3$					
L	$\uparrow 4$					
L	$\uparrow 5$					

- 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	$\leftarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\leftarrow 4$	$\leftarrow 5$
S	$\uparrow 1$	$\leftarrow\nwarrow\uparrow 2$	$\leftarrow\nwarrow\uparrow 3$	$\nwarrow\uparrow\leftarrow 4$	$\nwarrow\uparrow\leftarrow 5$	$\nwarrow\uparrow\leftarrow 6$
T	$\uparrow 2$	$\nwarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\leftarrow 4$	$\leftarrow 5$
A	$\uparrow 3$	$\uparrow 2$	$\nwarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\leftarrow 4$
L	$\uparrow 4$	$\uparrow 3$	$\uparrow 2$	$\leftarrow\nwarrow\uparrow 3$	$\nwarrow 2$	$\leftarrow 3$
L	$\uparrow 5$	$\uparrow 4$	$\uparrow 3$	$\leftarrow\nwarrow\uparrow 4$	$\nwarrow\uparrow 3$	$\leftarrow\nwarrow\uparrow 4$

- Exercises for you:
  - How many different optimal alignments are there?
  - Reconstruct all the optimal alignments.
  - Redo the chart with all costs $= 1$ (Levenshtein distance)

Alignment and MED: uses?

Computing distances and/or alignments between arbitrary strings can be used for

- Spelling correction (as here)
- Morphological analysis: which words are likely to be related?
- Other fields entirely: e.g., comparing DNA sequences in biology.
- Related algorithms are also used in speech recognition and timeseries data mining.

Getting rid of hand alignments

Using MED algorithm, we can now produce the character alignments we need to estimate our error model, given only corrected words.

- Previously, we needed hand annotations like:

actual:	n	o	-		m	u	u	c	h		e	f	f	e	r	t
	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
intended:	n	o	t		m	-	u	c	h		e	f	f	o	r	t

- Now, our annotation requires less effort:

actual:	no	muuch	effert
intended:	not	much	effort

Catch-22

- But wait! In my example, we used costs of 1 and 2 to compute alignments.

- We actually want to compute our alignments using the costs from our noise model: the most probable alignment under that model.

- But until we have the alignments, we can't estimate the noise model...

We'll deal with this Catch 22 next time!

Summary

- Minimum edit distance: a tractable way of computing the optimal sequence of operations for getting from one string to another.

- If you have accurate costs for each kind of operation

  – deletion, insertion, substitution

  then all you need is a set of unedited vs. edited documents to get the noise model.

- Together with the language model (trained on vast data), you have a spelling correction system!

- **Problem for next time:** how do you acquire estimates of the costs for each operation when you don't have annotated alignments?