

# Spelling correction

What string did the author intend to write?

# Model

How can we model the process of text generation to suggest a solution to this problem?

## Noisy-channel model

$$\hat{w} = \arg \max_{w \in V} P(w|x)$$

- $x$  is the observed word
- $V$  is the vocabulary (English dictionary)
- $\hat{w}$  is the corrected word

## Noisy-channel model, cont.

$$P(w|x) = \frac{P(x|w)P(w)}{P(x)}$$

- $P(x|w)$  is the "error model"
- $P(w)$  is the *a priori* probability of the author having written  $w$
- $P(x)$  can be considered a normalizing factor

$$\hat{w} = \arg \max_{w \in V} \frac{P(x|w)P(w)}{P(x)} = \arg \max_{w \in V} P(x|w)P(w)$$

## Error model

- $P(x|w) = \max_f P(f)$  where  $x = f(w)$
- $f(w) = g_0(g_1(\dots g_E(w)))$ 
  - corruption is the composition of  $E$  atomic "edits"
- $P(f) = \prod_{i=1..E} P(g_i)$ 
  - edits are independent
- $P(g_i) = p$ 
  - the probability of each edit is the same

## Minimum edit distance

$$\begin{aligned} P(x|w) &= \max_E \prod_{i=1..E} p \\ &= \max_E p^E \text{ s.t. } \exists x = f_E(w) \\ &= p^{\hat{E}} \end{aligned}$$

$$\hat{E} = \min E \text{ s.t. } \exists x = f_E(w)$$

$\hat{E}$  is the "minimum edit distance", the minimum number of edits necessary to convert  $w$  to  $x$ .

## Putting it all together

$$\begin{aligned}\hat{w} &= \arg \max_{w \in V} \log P(x|w) + \log P(w) \\ &= \arg \max_{w \in V} \hat{E} \log p + \log P(w)\end{aligned}$$

Note that with a non-uniform prior, you must choose  $p$ . It is effectively a weighting factor between the prior and the evidence.

# "Edit" operations

- insertion: `acress` → `actress`
- deletion: `acress` → `cross`
- substitution: `acress` → `access`
- transposition: `acress` → `caress`

Each operation is assigned a cost:

	LCS	Levenshtein	Damerau–Levenshtein
insertion	1	1	1
deletion	1	1	1
substitution	inf	1	1
transposition	inf	inf	1



# Longest common subsequence (LCS)

"subsequence": derived from another sequence by deleting elements

LCS *similarity*  $s$ :

- Levenshtein, listen  $\rightarrow$  lsten (5)
- access, aces  $\rightarrow$  aces (4)

LCS *distance*  $d = |x| + |w| - 2s$

We can transform from one to the other through  $d$  insertions/deletions (minimum).

# Levenshtein distance

Vladimir Levenshtein, 1965

The minimum number of insertions/deletions/substitutions required to transform string A into string B.

Note, this is a proper distance metric:

$$1. lev(A, B) = lev(B, A)$$

$$2. lev(A, C) \leq lev(A, B) + lev(B, C)$$

$$3. lev(A, B) = 0 \rightarrow A = B$$

## Algorithm

$$lev(A[:i], B[:j]) = \min \begin{cases} lev(A[:i], B[:j-1]) + 1 \\ lev(A[:i-1], B[:j]) + 1 \\ lev(A[:i-1], B[:j-1]) + \mathbf{1}_{A[i] \neq B[j]} \end{cases}$$

# Computation

- recursion?
  - just compute  $lev(A, B)$  according to the formula
  - ends up computing e.g.  $lev(A[: 1], B[: 1])$  many times...
- dynamic programming!
  - = recursion + memoization
  - or, tabular approach:
    - start computing at  $lev(A[: 0], B[: 0])$
    - increase  $i$  and  $j$ , using old results as needed
  - much more efficient

# Minimum edit distance

fill → still

one substitution (f → t) + one insertion (s) = 2

	_	s	t	i	l	l
f	0	1	2	3	4	5
i	1	1	2	3	4	5
l	2	2	2	2	3	4
l	3	3	3	3	2	3
l	4	4	4	4	3	2

## Damerau-Levenshtein distance

Levenshtein, plus transposition

- transposition is a common error when typing

note:  $lcs(A, B) \geq lev(A, B) \geq dlev(A, B)$

## Other applications of edit distance

- nearest-neighbor classification

## References

- Jurafsky and Martin, 2.5
- Jurafsky and Martin, B