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

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

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

Noisy-channel model, cont.

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

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

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

we're maximizing with respect to W that factor is shared by all of the terms so, we can just get rid of it

the model is tractable, computable and believable. but not true

Error model

- $P(x|w) = \max_f P(f)$ where x = f(w)
- ullet $f(w)=g_0(g_1(...g_E(w)))$ applying a sequence of edit
 - \circ corruption is the composition of E atomic "edits"

meaning: in some ways can be broken down

$$ullet P(f) = \prod_{i=1..E} P(g_i)$$

edits are independent
 the probability of this horrible thing happening is just the probability of each of the individual

$$\begin{array}{c} \bullet \ \ P(g_i) = p \\ \quad \circ \ \ \text{the probability of each edit is the same} \end{array}$$

wrong:edits are not independent (you place your hand on the keyboard and you star typing, you are making mistakes and they are correlated

Minimum edit distance

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

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

Putting it all together

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

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 → cress

• 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 → lsten (5)
- access, aces → 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.
$$lev(A,B)=0
ightarrow A=B$$

Algorithm

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

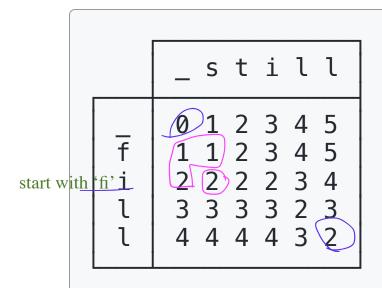
Computation

- recursion?
 - \circ just compute lev(A,B) according to the formula
 - \circ 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])
 - ullet increase i and j, using old results as needed
 - much more efficient

Minimum edit distance

fill → still

one substitution (f \rightarrow t) + one insertion (s) = 2



to compute the 2 in circle, we need information in the square"112"

Damerau-Levenshtein distance

Levenshtein, plus tranposition

• 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