Natural Language Processing

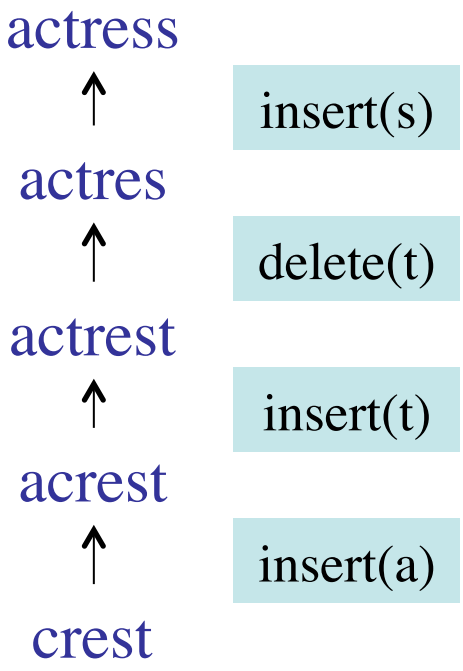
Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

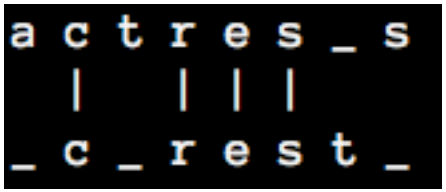
October 30, 2014

Minimum Cost Edit Distance

- Edit a source string into a target string
- Each edit has a cost
- Find the minimum cost edit(s)



Minimum Cost Edit Distance

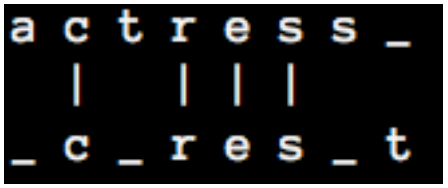


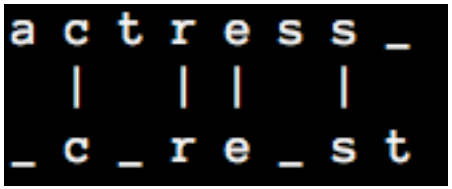
target

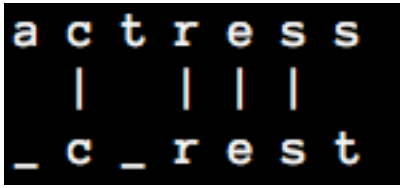
source

actress
↑
actres
↑
actrest
↑
acrest
↑
crest

minimum cost edit distance can be accomplished in multiple ways







Only 4 ways to edit source to target **for this pair**

Levenshtein Distance

- Cost is fixed across characters
  - Insertion cost is 1
  - Deletion cost is 1
- Two different costs for substitutions
  - Substitution cost is 1 (transformation)
  - Substitution cost is 2 (one deletion + one insertion)

Левенштейн Владимир

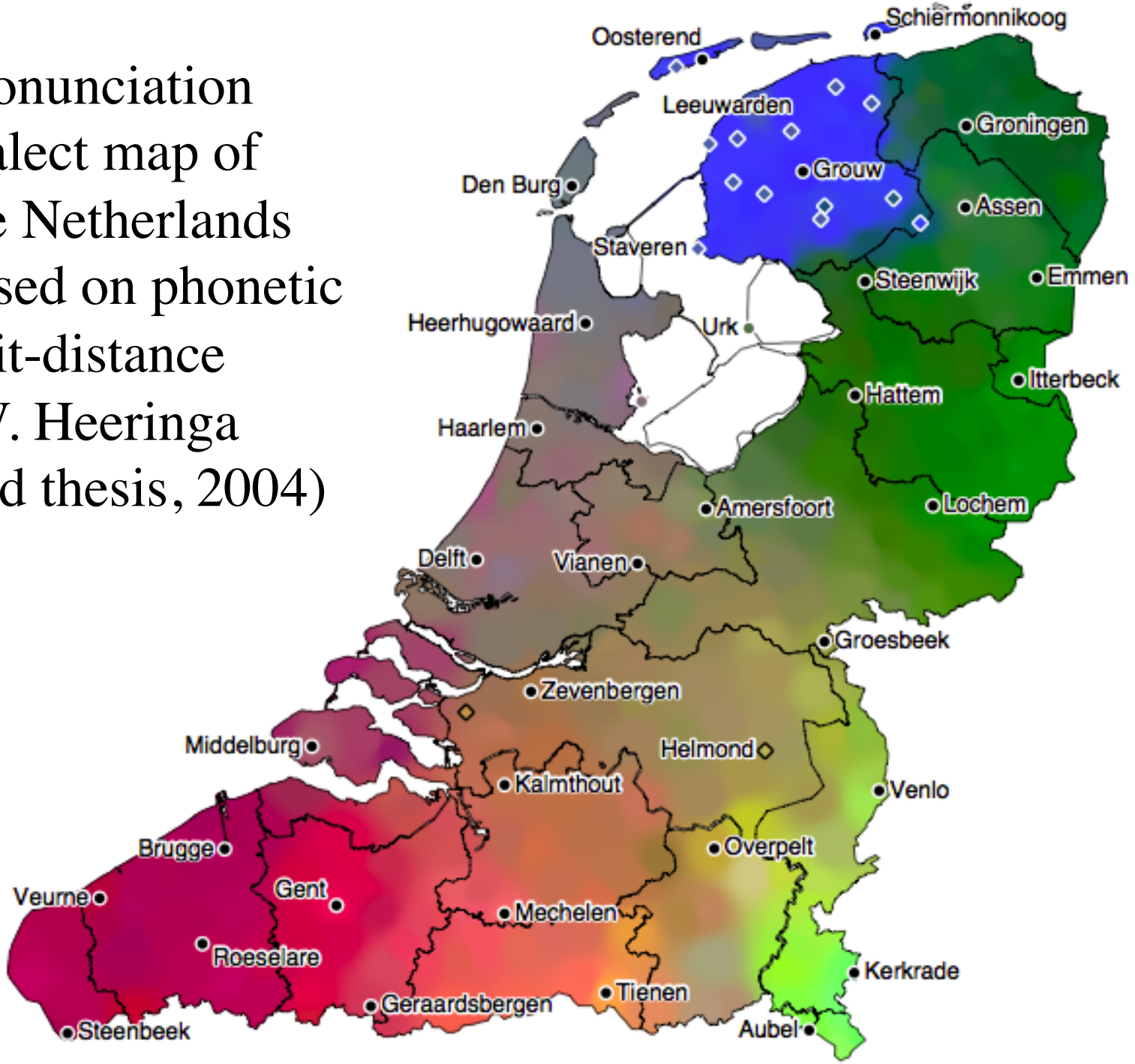
Vladimir Levenshtein

What's the edit distance?

Edit distance

- Useful in many NLP applications
- In some cases, we need edits with multiple characters, e.g. 2 chars deleted for one cost
- Comparing system output with human output, e.g. *input:* ibm *output:* IBM vs. Ibm (TrueCasing of speech recognition output)
- Error correction, e.g. spelling correction
- Defined over character edits or word edits, e.g. MT evaluation:
  - Foreign investment in Jiangsu ‘s agriculture on the increase
  - Foreign investment in Jiangsu agricultural investment increased

Pronunciation dialect map of the Netherlands based on phonetic edit-distance (W. Heeringa Phd thesis, 2004)



Consider two strings:

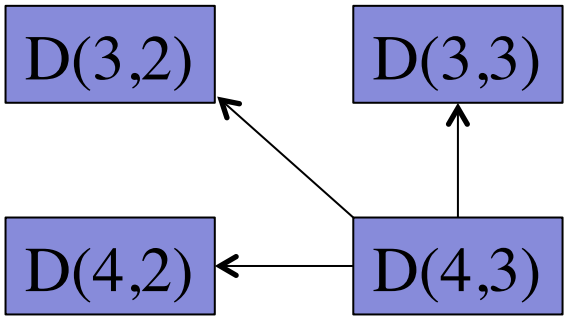
target = $g_1a_2m_3b_4l_5e_6$

source= $g_1u_2m_3b_4o_5$

- We want to find D(6,5)
- We find this recursively using values of D(i,j) where i≤6 j≤5
- For example, consider how to compute D(4,3)

target = $g_1a_2m_3b_4$

source= $g_1u_2m_3$



Take the minimum

- Case 1: SUBSTITUTE b_4 for m_3
- Use previously stored value for D(3,2)
- Cost($g_1a_2m_3b$ and g_1u_2m) = D(3,2) + cost(b≈m)
- For substitution: D(i,j) = D(i-1,j-1) + cost(subst)

- Case 2: INSERT b_4
- Use previously stored value for D(3,3)
- Cost($g_1a_2m_3b$ and $g_1u_2m_3$) = D(3,3) + cost(ins b)
- For substitution: D(i,j) = D(i-1,j) + cost(ins)

- Case 3: DELETE m_3
- Use previously stored value for D(4,2)
- Cost($g_1a_2m_3b_4$ and g_1u_2m) = D(4,2) + cost(del m)
- For substitution: D(i,j) = D(i,j-1) + cost(del)

Minimum Cost Edit Distance

- An alignment between target and source

Find $D(n,m)$ recursively

$$t_1, t_2, \ldots, \ldots, t_n$$
$$s_1, s_2, \ldots, s_m$$

$$D(i,j) = min \begin{cases} D(i-1,j) & +\text{cost}(t_i, \emptyset) \textit{ insertion into target} \\ D(i-1,j-1) & +\text{cost}(t_i, s_j) \textit{substitution/identity} \\ D(i,j-1) & +\text{cost}(\emptyset, s_j) \textit{deletion from source} \end{cases}$$

$$D(0,0) = 0$$

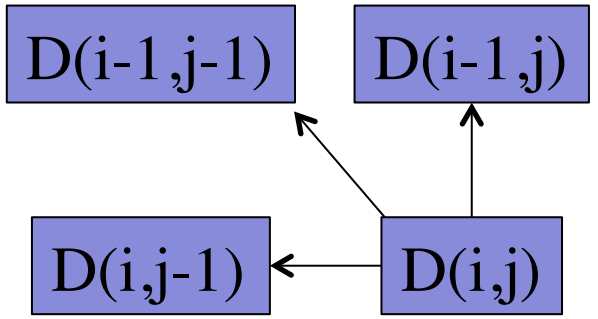
$$D(i,0) = D(i-1,0) + \text{cost}(t_i, \emptyset)$$

$$D(0,j) = D(0,j-1) + \text{cost}(\emptyset, s_j)$$

```
Function MinEditDistance (target, source)

 n = length(target)
 m = length(source)
 Create matrix D of size (n+1,m+1)
 D[0,0] = 0

 for i = 1 to n
   D[i,0] = D[i-1,0] + insert-cost

 for j = 1 to m
   D[0,j] = D[0,j-1] + delete-cost

 for i = 1 to n
   for j = 1 to m
     D[i,j] = MIN(D[i-1,j] + insert-cost,
                  D[i-1,j-1] + subst/eq-cost,
                  D[i,j-1] + delete-cost)
 return D[n,m]
```

D(i,j)

		g
	0	1
g	1	0



		g	u	m
	0	1	2	3
g	1	0	1	2
a	2	1	2	3
m	3	2	3	2
b	4	3	4	3

D(i,j)

		g	u	m	b	o
	0	1	2	3	4	5
g	1	0	1	2	3	4
a	2	1	2	3	4	5
m	3	2	3	2	3	4
b	4	3	4	3	2	3
l	5	4	5	4	3	4
e	6	5	6	5	4	5

Backtracing to find the alignments

D(i,j)

		g	u	m	b	o
	0	1	2	3	4	5
g	1	0 e	1	2	3	4
a	2	1	2 s	3	4	5
m	3	2	3	2 e	3	4
b	4	3	4	3	2 e	3
l	5	4	5	4	3 i	4
e	6	5	6	5	4	5 s

```
g a m b l e
|   | | |
g u m b _ o
```

Variable Cost Edit Distance

- So far, we have seen edit distance with uniform insert/ delete cost
- In different applications, we might want different insert/ delete costs for different items
- For example, consider the simple application of spelling correction
- Users typing on a qwerty keyboard will make certain errors more frequently than others
- So we can consider insert/delete costs in terms of a probability that a certain alignment occurs between the *correct* word and the *typo* word

Spelling Correction

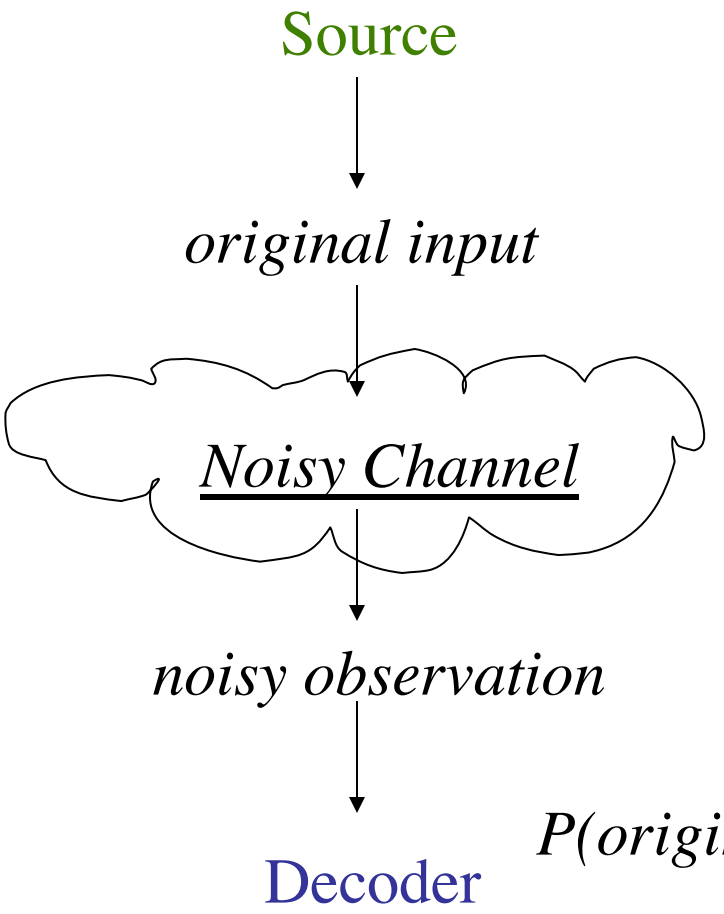
- Types of spelling correction
  - non-word error detection

    e.g. *hte* for *the*
  - isolated word error detection

    e.g. *acres* vs. *access* (cannot decide if it is the right word for the context)
  - context-dependent error detection (real world errors)

    e.g. *she is a talented acres* vs. *she is a talented actress*
- For simplicity, we will consider the case with exactly 1 error

Noisy Channel Model



Bayes Rule: *computing P(orig | noisy)*

- let x = *original input*, y = *noisy observation*

$$p(x \mid y) = \frac{p(x,y)}{p(y)} \qquad p(y \mid x) = \frac{p(y,x)}{p(x)}$$

$$p(x,y) = p(y,x)$$

$$p(x \mid y) \times p(y) = p(y \mid x) \times p(x)$$

$$p(x \mid y) = \frac{p(y \mid x) \times p(x)}{\cancel{p(y)}}$$ *Bayes Rule*

Single Error Spelling Correction

- Insertion (addition)
  - acress vs. cress
- Deletion
  - acress vs. actress
- Substitution
  - acress vs. access
- Transposition (reversal)
  - acress vs. caress

Noisy Channel Model for Spelling Correction (Kernighan, Church and Gale, 1990)

- *t* is the word with a single typo and *c* is the correct word

$$P(c \mid t) = p(t \mid c) \times p(c)$$

Bayes Rule

- Find the best candidate for the correct word

$$\hat{c} = \underset{c \in C}{\arg\max} P(t \mid c) \times P(c)$$

$$P(t \mid c) = ?? \qquad P(c) = \frac{f(c)}{N}$$

C is all the words in the vocabulary; |C| = N

Noisy Channel Model for Spelling Correction (Kernighan, Church and Gale, 1990) single error, condition on previous letter

P(*poton* | *potion*)

P(*poton* | *piton*)

$$P(t \mid c) = \begin{cases} \frac{del[c_{p-1},c_p]}{chars[c_{p-1},c_p]} & (xy)_c \text{ typed as } (x)_t \\ \frac{ins[c_{p-1},t_p]}{chars[c_{p-1}]} & (x)_c \text{ typed as } (xy)_t \\ \frac{sub[t_p,c_p]}{chars[c_p]} & (y)_c \text{ typed as } (x)_t \\ \frac{rev[c_p,c_{p+1}]}{chars[c_p,c_{p+1}]} & (xy)_c \text{ typed as } (yx)_t \end{cases}$$

t = *poton*
c = *potion*
del[t,i]=427
chars[t,i]=575
P = .7426

t = *poton*
c = *piton*
sub[o,i]=568
chars[i]=1406
P = .4039

Noisy Channel model for Spelling Correction

- The *del*, *ins*, *sub*, *rev* matrix values need data in which contain known errors

  (**training data**)

  e.g. Birbeck spelling error corpus

  (http://ota.ahds.ac.uk/texts/0643.html)
- Accuracy on single errors on unseen data

  (**test data**)

from (Kernighan, Church and Gale, 1990)

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																								
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Noisy Channel model for Spelling Correction

- Easily extended to multiple spelling errors in a word using edit distance algorithm
- Using learned costs for ins, del, replace
- Experiments: 87% accuracy for machine vs. 98% average human accuracy
- What are the limitations of this model?

*…was called a "stellar and versatile **acress** whose combination of sass and glamour has defined her …*

KCG model best guess is **acres**

More on spell checking

- Check out Peter Norvig's introduction to spell checking
  - http://norvig.com/spell-correct.html
- Better version of this appears in a book chapter
  - http://norvig.com/ngrams/