CMPT 413
Computational Linguistics

Anoop Sarkar

http://www.cs.sfu.ca/~anoop

Minimum Cost Edit Distance

- String edit distance: what is the minimum number of changes (char insertions or deletions) to transform the string *intention* into *execution* ?
- Assume cost of insertion is 1 and cost of deletion is 1
- Note that we assume that we can only change one character at a time

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)

Minimum Cost Edit Distance

- Algorithm using a Finite-state transducer:
  - construct a finite-state transducer with all possible ways to transduce *intention* (source = input) into *execution* (target = output)
  - We do this transduction one char at a time
  - A transition x:x gets zero cost and a transition on ε:x (insertion) or x:ε (deletion) for any char x gets cost 1
  - Finding minimum cost edit distance == Finding the shortest path from start state to final state

Edit Distance

- Think of it as an alignment between target and source

$t_1, t_2, \ldots, \ldots, t_n$

$s_1, s_2, \ldots, s_m$

Find *D(n,m)* recursively

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

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

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

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

```
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]
```

target

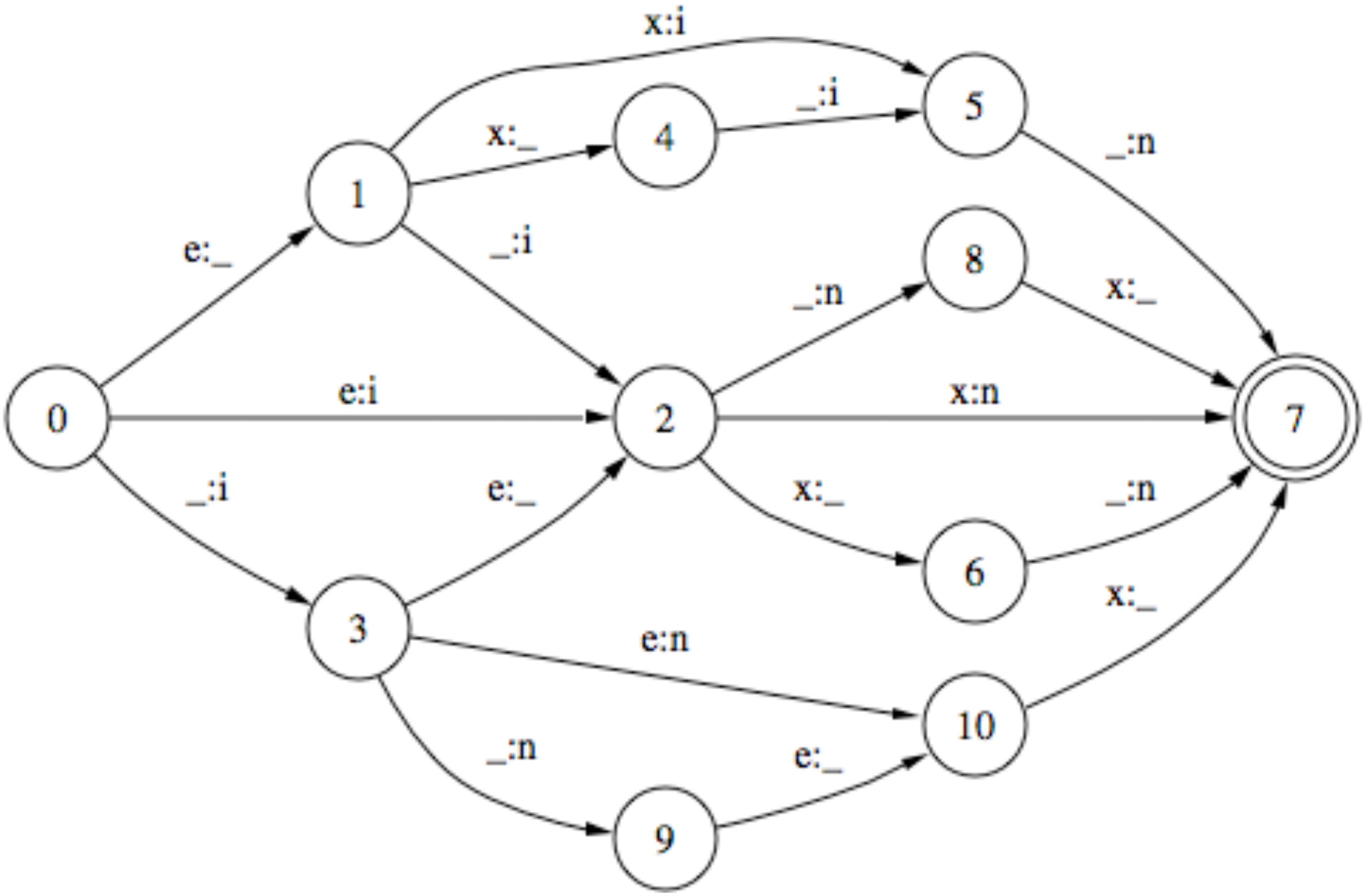
source

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

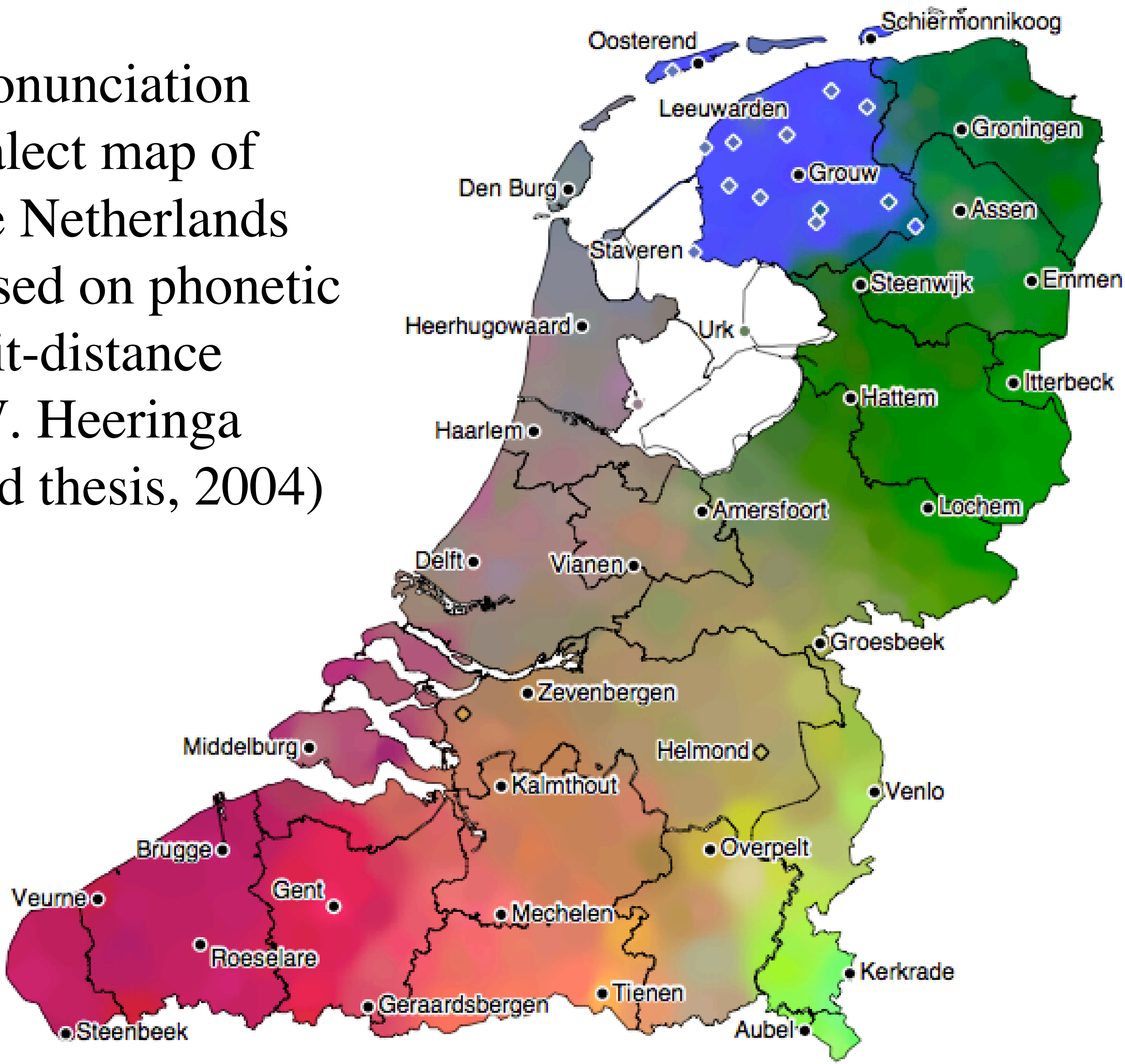
Edit distance

- Useful in many NLP applications
- In some cases, we need to generalize to 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
- Error 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

Edit distance and FSTs



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



Spelling Correction

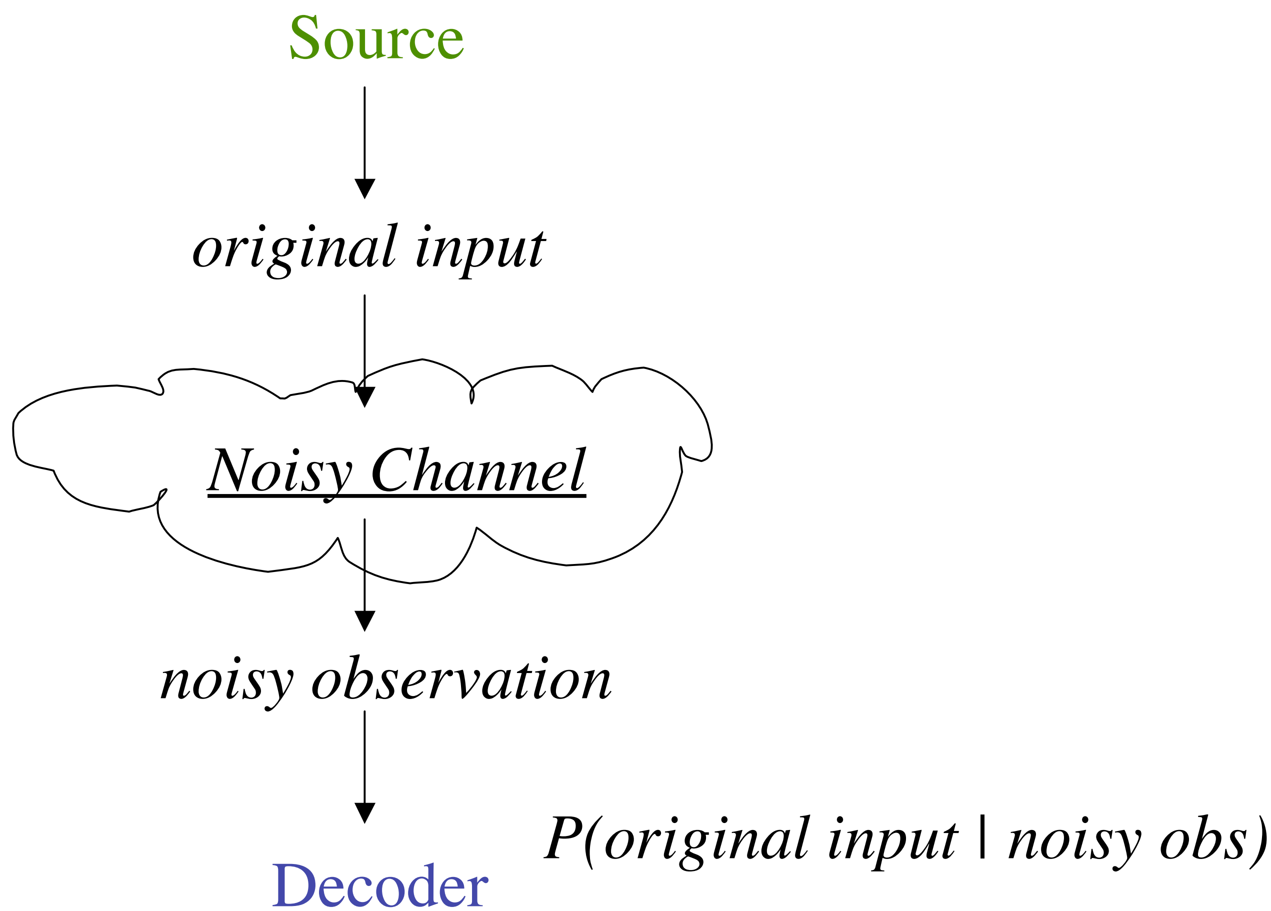
- Types of spelling correction
  - non-word error detection

    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*

Noisy Channel Model



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

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

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

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

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

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

<u>*Bayes Rule*</u>

Chain Rule

$$
\begin{aligned}
p(a,b,c \mid d) = \quad & p(a \mid b,c,d) \times \\
& p(b \mid c,d) \times \\
& p(c \mid d)
\end{aligned}
$$

Approximations: Bias vs. Variance

$$
\begin{aligned}
p(a \mid b,c,d) \approx \quad & p(a \mid b,c) \quad \textit{less } \textbf{\textit{bias}} \\
& p(a \mid b) \\
& p(a) \quad \textit{more } \textbf{\textit{variance}}
\end{aligned}
$$

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 typo and c is the correct word

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

- Find the best candidate for the correct word

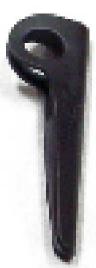
$$\hat{c} = \underset{c \in C}{\arg\max}\, P(t \mid c) \times P(c)$$

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

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

P(*poton* | *potion)*

P(*poton* | *piton)*



$$P(t \mid c) = \begin{cases} \dfrac{del[c_{p-1}, c_p]}{chars[c_{p-1}, c_p]} & (xy)_c \text{ typed as } (x)_t \\[2ex] \dfrac{ins[c_{p-1}, t_p]}{chars[c_{p-1}]} & (x)_c \text{ typed as } (xy)_t \\[2ex] \dfrac{sub[t_p, c_p]}{chars[c_p]} & (y)_c \text{ typed as } (x)_t \\[2ex] \dfrac{rev[c_p, c_{p+1}]}{chars[c_p, c_{p+1}]} & (xy)_c \text{ typed as } (yx)_t \end{cases}$$

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**)
- Accuracy on single errors on unseen data
  (**test data**)

Noisy Channel model for Spelling Correction

- 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**