LING 438/538
Computational Linguistics

Sandiway Fong
Lecture 17: 10/25
Last Time

- Prolog implementation of Finite State Transducers (FST) for morphological processing

- Let’s revisit the implementation of:

**(Context-Sensitive) Spelling Rule:** (3.5)

\[ \varepsilon \rightarrow e / \{x, s, z\}^\_ \# \]

\[
\begin{array}{c|c|c|c|c|c|c}
\hline
& f & o & x & +N & +PL & \\
\hline
\text{Lexical} & & & & & & \\
\hline
\text{Intermediate} & f & o & x & ^\# & s & \#
\text{left} & \text{right} & \\
\hline
\text{Surface} & f & o & x & e & s & \\
\hline
\end{array}
\]
Stage 2:

*Intermediate ⇔ Surface Levels*

- context-sensitive spelling rule also can be implemented using a FST

---

**Figure 3.14** The transducer for the E-insertion rule of (3.5), extended from a similar transducer in Antworth (1990).
FST 3.14 in Prolog

- % To run: ?- q0([f,o,x,^,s,#],L). L = [f,o,x,e,s,#]
- % q0
- q0([],[]).
- q0([z|L1],[z|L2]) :- !, q1(L1,L2). % q0 - z -> q1
- q0([s|L1],[s|L2]) :- !, q1(L1,L2). % q0 - s -> q1
- q0([x|L1],[x|L2]) :- !, q1(L1,L2). % q0 - x -> q1
- q0([#|L1],[#|L2]) :- !, q0(L1,L2). % q0 - # -> q0
- q0([^[L1],L2) :- !, q0(L1,L2). % q0 - ^:ep -> q0
- q0([X|L1],[X|L2]) :- q0(L1,L2). % q0 - other -> q0
FST 3.14 in Prolog

- % q1
- q1([],[]). % final state case
- q1([z|L1],[z|L2]) :- !, q1(L1,L2). % q1 - z -> q1
- q1([s|L1],[s|L2]) :- !, q1(L1,L2). % q1 - s -> q1
- q1([x|L1],[x|L2]) :- !, q1(L1,L2). % q1 - x -> q1
- q1([^[|L1],L2) :- !, q2(L1,L2). % q1 - ^:ep -> q2
- q1([#|L1],[#|L2]) :- !, q0(L1,L2). % q1 - # -> q0
- q1([X|L1],[X|L2]) :- q0(L1,L2). % q1 - other -> q0
% q2
q2([],[]).
q2([s|L1],[s|L2]) :- !, q5(L1,L2). % q2 - s -> q5
q2([z|L1],[z|L2]) :- !, q1(L1,L2). % q2 - z -> q1
q2([x|L1],[x|L2]) :- !, q1(L1,L2). % q2 - x -> q1
q2([#|L1],[#|L2]) :- !, q0(L1,L2). % q2 - # -> q0
q2([X|L1],[X|L2]) :- \+ X = ^, q0(L1,L2). % q2 - other -> q0
FST 3.14 in Prolog

- % q3
  - q3([s|L1],[s|L2]) :- q4(L1,L2). % q3 - s -> q4

- % q4
  - q4([#|L1],[#|L2]) :- q0(L1,L2). % q4 - # -> q0
FST 3.14 in Prolog

- % q5
- q5([z|L1],[z|L2]) :- !, q1(L1,L2). % q5 - z -> q1
- q5([s|L1],[s|L2]) :- !, q1(L1,L2). % q5 - s -> q1
- q5([x|L1],[x|L2]) :- !, q1(L1,L2). % q5 - x -> q1
- q5([^|L1],L2) :- !, q2(L1,L2). % q5 - ^:e -> q2
- q5([X|L1],[X|L2]) :- \+ X = #, q0(L1,L2). % q5 - other -> q0
Stage 2: *Intermediate ⇔ Surface Levels*

- **Prolog queries**
  
  \[ \text{Intermediate} \Rightarrow \text{Surface Level} \]
  
  - \(| ? - q0([f,o,x,^,s,#],L). \)
  - \(L = [f,o,x,e,s,#] ? \)
  - \(| ? - q0([c,a,t,^,s,#],L). \)
  - \(L = [c,a,t,s,#] ? \)
  - \(| ? - q0([s,i,s,t,e,r,^,s,#],L). \)
  - \(L = [s,i,s,t,e,r,s,#] ? \)

- **Prolog queries**
  
  \[ \text{Surface} \Rightarrow \text{Intermediate Level} \]
  
  - \(| ? - q0(X,[f,o,x,e,s,#]). \)
  - \(! \text{ Resource error: insufficient memory} \)
  - \((\text{Infinite loop}) \)
  - **Fix it!**
Stage 2:

Intermediate ⇔ Surface Levels

• Use the Prolog debugger...

• ?- trace.
• ?- notrace.
Today’s Topic

• Chapter 5
  – Spelling errors and correction
  – Error Correction
    • correct
      – Bayesian Probability
  – Minimum Edit Distance Computation
    • Dynamic Programming
Spelling Errors

• Textbook cites (Kukich, 1992):
  – *Non-word detection*  *(easiest)*
    • graffe  (giraffe)
  – *Isolated-word (context-free) error correction*
    • graffe  (giraffe,…)
    • graffed  (gaffed,…)
    • *by definition cannot correct when error word is a valid word*
  – *Context-dependent error detection and correction*  *(hardest)*
    • your an idiot  ⇒  you’re an idiot
    • *(Microsoft Word corrects this by default)*
Spelling Errors

• **OCR**
  – *visual similarity*
    - h⇔b, e⇔c, jump⇔jurnps

• **Typing**
  – *keyboard distance*
    - small⇔smsll, spell⇔spel;

• **Graffiti** (*many HCI studies*)
  – *stroke similarity*
    - Common error characters are: V, T, 4, L, E, Q, K, N, Y, 9, P, G, X
    - Two stroke characters: B, D, P (error: two characters)

• **Cognitive Errors**
  – *bad spellers*
    - separate⇔seperate
correct

- textbook section 5.5
  - *we will go through again later in more detail when we do chapter 6*

- Kernighan et al. (correct)
  - take typo *t* (not a word)
    - mutate *t* minimally by deleting, inserting, substituting or transposing (swapping) a letter
    - look up “mutated *t*” in a dictionary
    - candidates are “mutated *t*” that are real words

- example (5.2)
  - *t* = across
  - *C* = {actress, cress, caress, access, across, acres, acres}
• **formula**
  - \( \hat{c} = \arg\max_{c \in C} P(t|c) P(c) \) \hspace{1em} (*Bayesian Inference*)
  - \( C = \{\text{actress, cress, caress, access, across, acres, acres}\} \)

• **Prior:** \( P(c) \)
  - *estimated using frequency information over a large corpus (N words)*
  - \( P(c) = \frac{\text{freq}(c)}{N} \)
  - \( P(c) = \frac{\text{freq}(c)+0.5}{(N+0.5V)} \)
    - avoid zero counts (*non-occurrences*)
    - (add fractional part 0.5)
    - *add one (0.5) smoothing*
      - see chapter 6
    - \( V \) is vocabulary size of corpus

<table>
<thead>
<tr>
<th>c</th>
<th>freq(c)</th>
<th>p(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>actress</td>
<td>1343</td>
<td>.0000315</td>
</tr>
<tr>
<td>cress</td>
<td>0</td>
<td>.000000014</td>
</tr>
<tr>
<td>caress</td>
<td>4</td>
<td>.0000001</td>
</tr>
<tr>
<td>access</td>
<td>2280</td>
<td>.000058</td>
</tr>
<tr>
<td>across</td>
<td>8436</td>
<td>.00019</td>
</tr>
<tr>
<td>acres</td>
<td>2879</td>
<td>.000065</td>
</tr>
</tbody>
</table>
• **Likelihood:** $P(t|c)$  \textit{probability of typo $t$ given candidate word $c$}
  
  using some corpus of errors
  
  compute following 4 \textit{confusion matrices}
  
  $\text{del}[x,y] = \text{freq(correct } xy \text{ mistyped as } x)$
  
  $\text{ins}[x,y] = \text{freq(correct } x \text{ mistyped as } xy)$
  
  $\text{sub}[x,y] = \text{freq(correct } x \text{ mistyped as } y)$
  
  $\text{trans}[x,y] = \text{freq(correct } xy \text{ mistyped as } yx)$
  
  $P(t|c) = \text{del}[x,y]/f(x)$ if $c$ related to $t$ by deletion of $y$
  
  $P(t|c) = \text{ins}[x,y]/f(x)$ if $c$ related to $t$ by insertion of $y$  \textit{etc…}

| c         | freq(c) | p(c)     | p(t|c)   | p(t|c)p(c) | %   |
|-----------|---------|----------|----------|------------|-----|
| actress   | 1343    | .0000315 | .000117  | $3.69 \times 10^{-9}$ | 37% |
| cress     | 0       | .000000014 | .00000144 | $2.02 \times 10^{-14}$ | 0%  |
| caress    | 4       | .0000001 | .00000164 | $1.64 \times 10^{-13}$ | 0%  |
| access    | 2280    | .000058  | .000000209 | $1.21 \times 10^{-11}$ | 0%  |
| across    | 8436    | .00019   | .00000093 | $1.77 \times 10^{-9}$ | 18% |
| acres     | 2879    | .000065  | .0000321 | $2.09 \times 10^{-9}$ | 21% |
| acres     | 2879    | .000065  | .0000342 | $2.22 \times 10^{-9}$ | 23% |
correct

• example
  – t = across
  – ^c = acres (44%)

• despite all the math
• wrong result for
  – was called a stellar and versatile across

| c    | freq(c) | p(c)    | p(t|c)   | p(t|c)p(c) | %    |
|------|---------|---------|---------|-----------|------|
| actress | 1343   | .0000315 | .000117 | 3.69 × 10^{-9} | 37%  |
| cress  | 0       | .00000014| .0000144 | 2.02 × 10^{-14} | 0%   |
| caress | 4       | .0000164 |         | 1.64 × 10^{-13} | 0%   |
| access | 2280    | .000058  | .00000209| 1.21 × 10^{-11} | 0%   |
| across | 8436    | .00019   | .000093 | 1.77 × 10^{-9}  | 18%  |
| acres  | 2879    | .000065  | .0000321 | 2.09 × 10^{-9}  | 21%  |
| acres  | 2879    | .000065  | .0000342 | 2.22 × 10^{-9}  | 23%  |

• what does Microsoft Word use?
  – was called a stellar and versatile across
Minimum Edit Distance

- textbook section 5.7
- general string comparison
- edit operations are insertion, deletion and substitution
- not just limited to distance defined by a single operation away (correct)
- we can ask how different is string $a$ from $b$ by the *minimum edit distance*

| Trace  | inten$\underline{t}$ion
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>///\   / / / /</td>
</tr>
<tr>
<td></td>
<td>ex$\underline{e}$cution</td>
</tr>
</tbody>
</table>

| Alignment | inten$\underline{s}$tion
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\underline{e}$x$\underline{e}$cution</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation List</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete i $\rightarrow$ inten$\underline{t}$ion</td>
</tr>
<tr>
<td>substitute n by e $\rightarrow$ ntention</td>
</tr>
<tr>
<td>substitute t by x $\rightarrow$ etention</td>
</tr>
<tr>
<td>insert u $\rightarrow$ exention</td>
</tr>
<tr>
<td>substitute n by c $\rightarrow$ exen$\underline{u}$tion</td>
</tr>
<tr>
<td>exen$\underline{u}$tion</td>
</tr>
<tr>
<td>$\underline{e}$xen$\underline{u}$tion</td>
</tr>
<tr>
<td>$\underline{e}$xen$\underline{u}$tion</td>
</tr>
</tbody>
</table>

*Figure 5.4* Three methods for representing differences between sequences (after Kruskal (1983))
Minimum Edit Distance

• **applications**
  – could be used for multi-typo correction
  – used in Machine Translation Evaluation (MTEval)
  – **example**
    • **Source**: 生産工程改善について
    • **Translations**:
      • (Standard) For improvement of the production process
      • (MT-A) About a production process betterment
      • (MT-B) About the production process improvement
    • **method**
      – compute edit distance between MT-A and Standard and MT-B and Standard in terms of word insertion/substitution etc.
Minimum Edit Distance

- **cost models**
  - **Levenshtein**
    - insertion, deletion and substitution all have unit cost
  - **Levenshtein (alternate)**
    - insertion, deletion have unit cost
    - substitution is twice as expensive
    - substitution = one insert followed by one delete
  - **Typewriter**
    - insertion, deletion and substitution all have unit cost
    - modified by key proximity
Minimum Edit Distance

• **Dynamic Programming**
  – divide-and-conquer
    • to solve a problem we divide it into sub-problems
  – sub-problems may be repeated
    • don’t want to re-solve a sub-problem the 2nd time around
  – idea: put solutions to sub-problems in a table
    • and just look up the solution 2nd time around, thereby saving time
    • *memoization*
Fibonacci Series

- **Definition**
  - $F(0) = 0$
  - $F(1) = 1$
  - $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$

- **Sequence**
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946...

- **Computation**
  - $F(5)$
  - $F(4) + F(3)$
  - $F(3) + F(2) + F(3)$
  - $F(2) + F(1) + F(2) + F(2) + F(1)$
Minimum Edit Distance

- can be defined incrementally
- example:
  - intention
  - execution
- suppose we have the following minimum costs
  - insertion, deletion have unit cost, substitution is twice as expensive
  - intent ⇒ execut (9)
  - inten ⇒ execut (9)
  - inten ⇒ execut (8)
- Minimum cost for intent ⇒ execut (8) is given by computing the possibilities
  - intent ⇒ execut (9) ⇒ execut (insert t) (10)
  - intent (delete t) ⇒ inten ⇒ execut (9) (10)
  - inten ⇒ execut (8): substitute t for t (zero cost) (8)

one edit operation away
Minimum Edit Distance

- Generally

\[
P(t|c) = \min \left\{ \begin{array}{l}
distance[i - 1, j] + \text{ins-cost}(\text{target}_i) \\
distance[i - 1, j - 1] + \text{subst-cost}(\text{source}_j, \text{target}_i) \\
distance[i, j - 1] + \text{del-cost}(\text{source}_j)
\end{array} \right. 
\]

<table>
<thead>
<tr>
<th>n</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

# execution
Minimum Edit Distance

- Generally

\[ P(t|c) = \min \left\{ \begin{array}{l}
    \text{distance}[i-1, j] + \text{ins-cost}(\text{target}_i) \\
    \text{distance}[i-1, j-1] + \text{subst-cost}(\text{source}_j, \text{target}_i) \\
    \text{distance}[i, j-1] + \text{del-cost}(\text{source}_j) \\
\end{array} \right\} \]

<table>
<thead>
<tr>
<th>n</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>i</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>t</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>n</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>e</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>n</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>i</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Example for \textit{inte} \leftrightarrow \textit{e}

Minimum cost should be 3
Minimum Edit Distance Computation

- Microsoft Excel implementation of the formula
- (from course webpage): try it!

$ in a cell reference means don’t change when copied from cell to cell
e.g. in C$1
1 stays the same in $A3
A stays the same