CS674 Natural Language Processing

- Last class
  - Spelling correction
  - Noisy channel model
  - Bayesian approach to spelling correction

- Today
  - Likelihood computation for spelling correction
  - Minimum edit distance
  - Bayesian method for pronunciation

Noisy channel model

- Channel introduces noise which makes it hard to recognize the true word.

- **Goal**: build a model of the channel so that we can figure out how it modified the true word...so that we can recover it.

---

Bayesian spelling correction

- Let \( c \) range over the set \( C \) of candidate corrections
- Let \( t \) represent the typo
- Select the most likely correction:

\[
\hat{c} = \arg \max_{c \in C} \frac{P(t | c) \cdot P(c)}{P(t)}
\]

---

Computing the prior

- Solution: *smoothing*

\[
P(c) = \frac{C(c) + 0.5}{N + 0.5 |V|}
\]
Computing the likelihood

- Computing the likelihood term $P(t|c)$ exactly is an unsolved problem.
- Can estimate its value:
  - The most important factors predicting an insertion, deletion, transposition are simple local factors.
- Simple method: estimate the number of times that a single-letter error occurs in some large corpus of errors.
  - E.g. estimate $P(\text{acress} | \text{across})$ using the number of times that e was substituted for o.

Confusion matrices

- One for each type of single-error:
  - $\text{sub}[x,y]$:
    - # of times that $x$ was typed as $y$.
    - $\text{sub}[o,e]$: # of times that e was substituted for o.
  - $\text{trans}[x,y]$:
    - # of times that $xy$ was typed as $yx$.
  - $\text{del}[x,y]$:
    - # of times that the characters $xy$ in the correct word were typed as $x$.
  - $\text{ins}[x,y]$:
    - # of times that the character $x$ in the correct word was typed as $xy$.

Estimating $P(t|c)$

- If deletion, e.g.
  
  $P(\text{acress}|\text{actress}) = \frac{\# \text{ times ct is mistyped as c}}{\# \text{ times ct appears}}$

- More generally,
  
  $P(t | c) = \frac{\text{del}[c_{p-1,c_p}]}{\text{count}(c_{p-1,c_p})}$

  where $c_p$ is the $p$th character of the word $c$.
  $t_p$ is the $p$th character of the word $t$.

Estimating $P(t|c)$

- If substitution, e.g.
  
  $P(\text{acress}|\text{across}) = \frac{\# \text{ times e is substituted for o}}{\# \text{ times o appears}}$

- More generally,
  
  $P(t | c) = \frac{\text{sub}[t_p,c_p]}{\text{count}(c_p)}$

  where $c_p$ is the $p$th character of the word $c$.
  $t_p$ is the $p$th character of the word $t$. 
Estimating $P(t|c)$

$$P(t|c) = \begin{cases} \text{del}(c_{p-1}, c_p)/\text{count}(c_{p-1}, c_p) & \text{if deletion} \\ \text{ins}(c_{p-1}, t_p)/\text{count}(c_{p-1}) & \text{if insertion} \\ \text{sub}(t_p, c_p)/\text{count}(c_p) & \text{if substitution} \\ \text{trans}(c_{p-1}, c_{p+1})/\text{count}(c_p, c_{p+1}) & \text{if transposition} \end{cases}$$

where $c_p$ is the $p$th character of the word $c$ and $t_p$ is the $p$th character of the word $t$.

Final probabilities

<table>
<thead>
<tr>
<th>c</th>
<th>freq(c)</th>
<th>p(c)</th>
<th>p(k)</th>
<th>p(k)p(c)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>actress</td>
<td>1343</td>
<td>0.00003145</td>
<td>0.00017</td>
<td>3.69 x 10^{-6}</td>
<td>53%</td>
</tr>
<tr>
<td>across</td>
<td>0</td>
<td>0.00000144</td>
<td>0.000144</td>
<td>2.02 x 10^{-6}</td>
<td>0%</td>
</tr>
<tr>
<td>access</td>
<td>4</td>
<td>0.00000164</td>
<td>0.000164</td>
<td>1.64 x 10^{-6}</td>
<td>0%</td>
</tr>
<tr>
<td>across</td>
<td>8436</td>
<td>0.0000058</td>
<td>0.000058</td>
<td>1.21 x 10^{-6}</td>
<td>0%</td>
</tr>
<tr>
<td>access</td>
<td>2879</td>
<td>0.00000321</td>
<td>0.000321</td>
<td>2.09 x 10^{-6}</td>
<td>2%</td>
</tr>
<tr>
<td>across</td>
<td>2879</td>
<td>0.00000342</td>
<td>0.000342</td>
<td>2.22 x 10^{-6}</td>
<td>23%</td>
</tr>
</tbody>
</table>

Context: …was called a “stellar and versatile across whose combination of sass and glamour has defined her”…

Assigning costs

- **Levenshtein distance**
  - cost (del) = cost (ins) = cost (subst) = 1
  - So the Levenshtein distance between *intention* and *execution* is 5

- Other common options
  - cost (del) = cost (ins) = 1
  - cost (subst) = 2
    - Because it counts as a deletion and an insertion

- Weight by more complex functions
  - E.g. using the confusion matrices discussed earlier

Spelling correction with multiple errors

- **computing string distance**
- E.g. use the **minimum edit distance** algorithm (Wagner and Fischer, 1974)
  - Determines the minimum number of editing operations (insertion, deletion, substitution) needed to transform one string into another
Computing minimum edit distance

- Use dynamic programming
- Intuition of dynamic programming solution is that a large problem can be solved by properly combining the solutions to various subproblems
- Operate by creating an edit-distance matrix
  - edit-distance[i,j] contains the distance between the first $i$ characters of the target and the first $j$ characters of the source

**min-edit-distance algorithm**

```python
function Min-Edit-Distance(target, source) returns min-distance

m ← LENGTH(target)

n ← LENGTH(source)

Create a distance matrix distance[i][j] for i from 1 to m+1

distance[0][0] ← 0

for each row i from 1 to m do
  for each column j from 1 to n do
    distance[i][j] ← Min (distance[i][j-1] + ins-cost(target[i]),
                            distance[i-1][j] + del-cost(source[j]),
                            distance[i-1][j-1] + subst-cost(source[j], target[i]))
```