Catching Typos

- Recognizing misspellings easy - check dictionary
- But lots of suffixes and prefixes: use fsa!
- What about making corrections to isolated words?
  - Look for spellings that are “close” to word
- Context-dependent errors detection/correction
  - Transpositions may accidentally create real words!

Spelling Correction

- Map words into equivalence classes that likely hold correct spelling.
- CS 51 lab: canonize words by removing vowels and doubled consonants: canonize lab
- Find all words w/same canonization as word.
- Alternatively, develop metric and find real world closest to word.
- Use minimum edit distance

Minimum Edit Distance

- Can convert any word to another by series of additions, deletions, and substitutions.
- Once specify cost of each operation then can measure distance between them
- We’ll use 1 for cost of addition/deletion, 2 for substitution.
- Use same algorithm if choose different costs, but get different answer.

Example

- Convert “INTENTION” to “EXECUTION”
  - INTENTION

Example

- Convert “INTENTION” to “EXECUTION”
  - INTENTION  delete 1
  - cost = 1
Example

- Convert “INTENTION” to “EXECUTION”
  - ETENTION subst E for N
    - Cost = 3

Example

- Convert “INTENTION” to “EXECUTION”
  - EXENTION subst X for T
    - Cost = 5

Example

- Convert “INTENTION” to “EXECUTION”
  - EXECTION subst C for N
    - Cost = 7

Example

- Convert “INTENTION” to “EXECUTION”
  - EXECUTION insert U
    - Cost = 8

Optimality

- How can we know if that is the minimal edit distance?
- Check all possible conversions? Too many!

Solving Problems

- Optimal substructure property: The optimal solutions to a problem contain optimal solutions to its subproblems.
- Ex: Shortest distance from LA to NYC
  - If shortest path goes through Chicago then portion of the path from LA to Chicago and from Chicago to NYC are also optimal.
**Dynamic Programming**

- If problem has overlapping subproblems (solve same problem repeatedly) & optimal substructure property then can use dynamic programming.
- Key idea is to save solutions to subproblems so don’t have to recalculate
  - Memoization!
- Can do top-down or bottom-up
  - We’ll do bottom-up

**Minimum Edit Distance**

- What is minimal cost of transforming v to w?
- Transform to problem with subproblems.
- Define distance[i,j] to be min cost of transforming v[1..i] to w[1..j]
- Does it satisfy optimal substructure property?
- Does it have overlapping subproblems?

**Minimum Edit Distance**

- Recursive solution:
  \[
  \text{distance}[i,j] = \begin{cases} 
  \text{ins}_\text{cost}(w_i) & \text{if } i > 0 \text{ and } j > 0 \\
  \text{distance}[i-1,j] + 1 & \text{if } i > 0 \\
  \text{distance}[i,j-1] + 1 & \text{if } j > 0 \\
  \text{distance}[i-1,j-1] + \text{sub}_\text{cost}(w_i, v_j) & \text{if } i > 0 \text{ and } j > 0 \\
  \end{cases}
  \]
  
  \text{ins}_\text{cost} = 1 \\
  \text{del}_\text{cost} = 1 \\
  \text{sub}_\text{cost} = 0 \text{ if } w_i = v_j \text{, otherwise 2}

**DISTANCE[1,J]**

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>A</th>
<th>R</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>P</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>N</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>K</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

*Recover edits from table*
**Variants & Improvements**

- Needleman-Wunch distance: cost of substitution varies depending on characters
  - E.g., distance between characters nearby is less
- Want to match names: Kim Barry Bruce, Kim B. Bruce, K. B. Bruce, Kim Bruce, K. Bruce.
  - One idea: n character gap costs less than n gaps of length 1.

**N-Grams**

- N-gram is sequence of N words that occur sequentially in text
- Determine probabilities of N-grams
- Use to predict which word is most likely to be correct in context.
- Can help in spelling correction

**Using context**

- Spell-checking:
  - They are leaving in about 15 minuets.
- Part of speech tagging
  - Which meaning of “dogs”
- Machine translation
- Speech & handwriting recognition
- Compare possible word decodings
- Authorship identification

**Which is most probable?**

- First Example:
  - ... I think they’re OK ...
  - ... I think there OK ...
  - ... I think their OK ...
- Second Example:
  - ... by the way, are they’re likely to ...
  - ... by the way, are there likely to ...
  - ... by the way, are their likely to ...

- Third Example:
  - How do you wreak a nice beach?
  - How do you recognize speech?
- Fourth Example:
  - Put the file in the folder
  - Put the file and the folder
**Counting Words**

- Types vs Tokens
  - "They picnicked by the pool, then lay back on the grass and looked at the stars"
  - 16 tokens, 14 types
  - Shakespeare: 884,647 tokens, 29,006 types
  - Also interested in number of lemmas
  - Remove affixes

**Language Models**

- Develop a “language model” to help us predict the likelihood of strings.
- In English:
  - \( P(\text{the big dog}) > P(\text{dog big the}) > P(\text{dog the which}) \)
- How can the computer know this?
- Each sentence is sequence \( w_1, ..., w_n \)
- How determine \( P(w_1, ..., w_n) \)

**N-Grams**

- Computes a probability for observed input
- Probability is likelihood of observation being generated by same source as training data.
- Different models arise from different training sets: English vs. French
- Problems!

**Any Questions?**