## CS 181:

#### NATURAL LANGUAGE PROCESSING

Lecture 4: Dynamic Programming & Minimum String Edit Distance

KIM BRUCE POMONA COLLEGE SPRING 2008

Disclaimer: Slide contents borrowed from many sources on web!

## CATCHING TYPOS

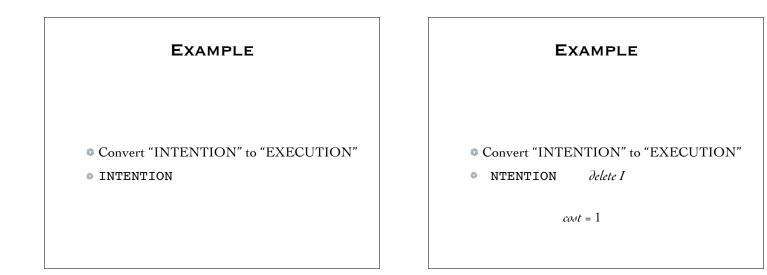
- Recognizing misspellings easy check dictionary
  - But lots of suffixes and prefixes: use fsa!
- What about making corrections to isolated words?
  - Look for spellings that are "close" to word
- Context-dependent errors detection/correction
   Transpositions may accidentally create real words!

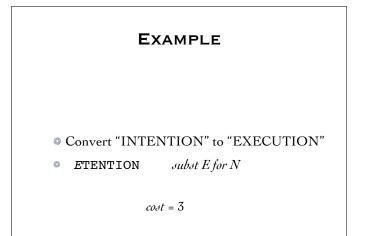
#### SPELLING CORRECTION

- Map words into equivalence classes that likely hold correct spelling.
  - CS 51 lab: canonize words by removing vowels and doubled consonants: <u>canonize lab</u>
  - 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.

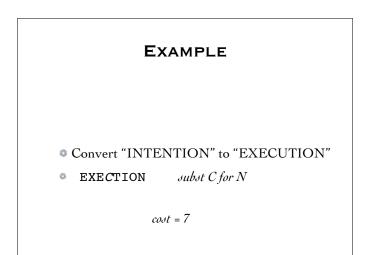


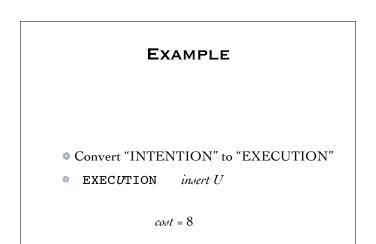


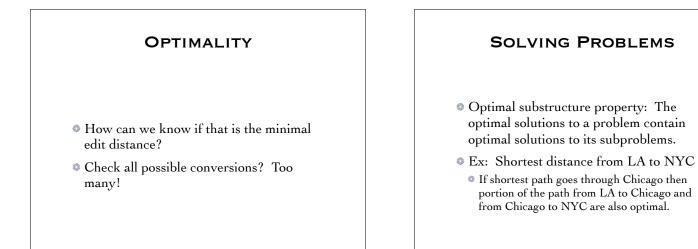
## EXAMPLE

- Convert "INTENTION" to "EXECUTION"
- \* **EXENTION** subst X for T

cost = 5







#### **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-upWe'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..j] to w[1..i]
- Does it satisfy optimal substructure property?
- Does it have overlapping subproblems?

### MINIMUM EDIT DISTANCE

Let |v| = m, |w| = n

#### Consider last move in aligning. 3 choices:

- Add move: Take moves changing v to w[1:n-1] & insert w[n]
- Delete move: Take moves changing v[1:m-1] to w & delete v[m]
- Replace move: Take moves changing v[1:m-1] to w[1:n-1] & change v[m] to w[n]

## MINIMUM EDIT DISTANCE

Recursive solution:

 $distance[i,j] = min \begin{cases} distance[i-1,j] + ins\_cost(w_i) \\ distance[i-1,j-1] + sub\_cost(w_i,v_j) \\ distance[i,j-1] + del\_cost(v_j) \end{cases}$ 

ins\_cost = 1 del\_cost = 1 sub\_cost = 0 if  $w_i = v_j$ , 2 otherwise

D	0IS7	ΓAN	ICE	:[],_	[]	
	#	Р	А	R	Κ	
#	0 +	-1 t	-2	- 3 ←	- 4	
S	1 +	$\geq \frac{1}{2} \leftarrow$	-3 ←	-4 ←	$-\frac{1}{5}$	
Р	2		- 2 ←	- 3 ←	- 4	
А	3	2	1.	- 2 ←	- 3	
Ν	4	-3			- 4	
K	$\frac{1}{5}$	4	3 ←	$\geq 4$	3	
	Rea	over edi	ts from	table		-

def minEditDist(target, source): n = len(target) m = len(source)
# <i>m+1 rows, n+1 cols</i> distance = [[0 for i in range(n+1)] for j in range(m+1)]
for col in range(1,n+1): distance[0][col] = distance[0][col-1] + 1 for row in range(1,m+1): distance[row][0]= distance[row-1][0] + 1
<pre>for col in range(1,n+1):     for row in range(1,m+1):         distance[row][col] = min(             distance[row-1][col] + 1,             distance[row][col-1] + 1,             distance[row-1][col-1]+substCost(source[row-1],target[col-1])) return distance[m][n]</pre>

#### VARIANTS & IMPROVEMENTS

- Needleman-Wunch distance: cost of substitution varies depending on characters
  - E.g., distance btn 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-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 ...

#### WHICH IS MOST PROBABLE?

- Third Example:
  - How do you wreck 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(the big dog) > P(dog big the) > P(dgo gib eth)
- \* 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?**