

# CS 562: Empirical Methods in Natural Language Processing

## Unit 1: Sequence Models

Lectures 11-13: Stochastic String Transformations  
(a.k.a. “channel-models”)

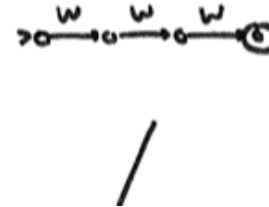
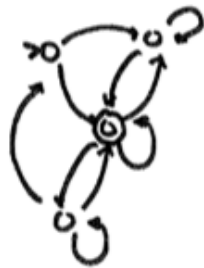
Weeks 5-6 -- Sep 29, Oct 1 & 6, 2009

Liang Huang ([lihuang@isi.edu](mailto:lihuang@isi.edu))

# String Transformations

- General Framework for many NLP problems
- Examples
  - Part-of-Speech Tagging
  - Spelling Correction (Edit Distance)
  - Word Segmentation
  - Transliteration, Sound/Spelling Conversion, Morphology
  - Chunking (Shallow Parsing)
  - Beyond Finite-State Models (i.e., tree transformations)
    - Summarization, Translation, Parsing, Information Retrieval, ...
- Algorithms: Viterbi (both max and sum)

# Review of Noisy-Channel Model



Application	Input	Output	$p(i)$	$p(o i)$
Machine Translation	$L_1$ word sequences	$L_2$ word sequences	$p(L_1)$ in a language model	translation model
Optical Character Recognition (OCR)	actual text	text with mistakes	prob of language text	model of OCR errors
Part Of Speech (POS) tagging	POS tag sequences	English words	prob of POS sequences	$p(w t)$
Speech recognition	word sequences	speech signal	prob of word sequences	acoustic model

# Example 1: Part-of-Speech Tagging

$$P(t \dots t \mid w \dots w)$$

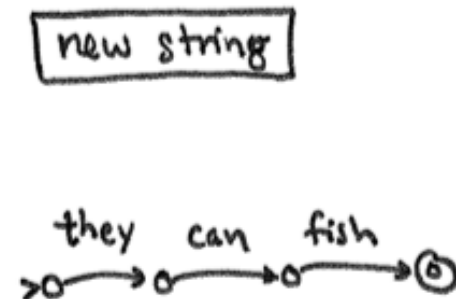
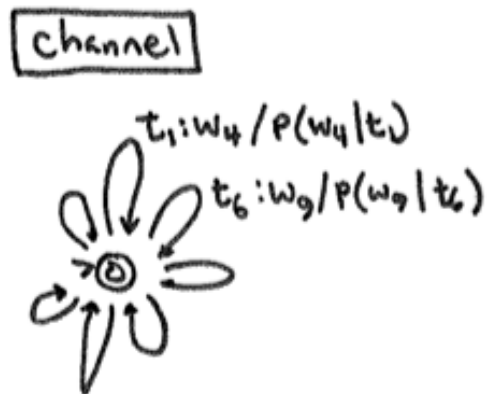
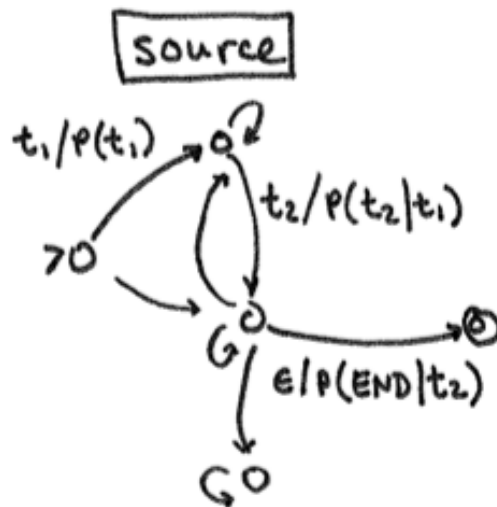
$$\sim P(t \dots t) \cdot P(w \dots w \mid t \dots t)$$

$$\sim \underbrace{P(t_1) \cdot P(t_2 \mid t_1) \dots P(t_n \mid t_{n-1})}_{\text{local grammar preference}} \cdot \underbrace{P(w_1 \mid t_1) \dots P(w_n \mid t_n)}_{\text{lexical preference}}$$

local grammar preference

lexical preference

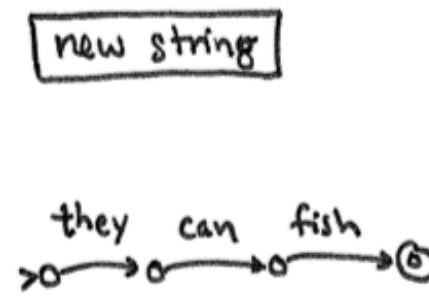
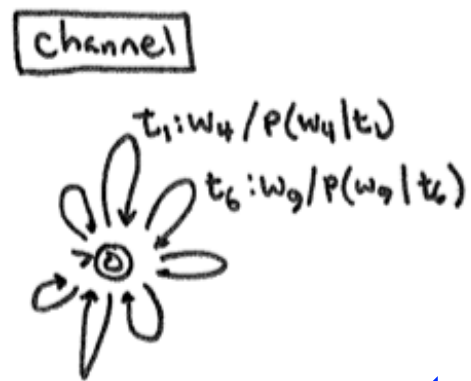
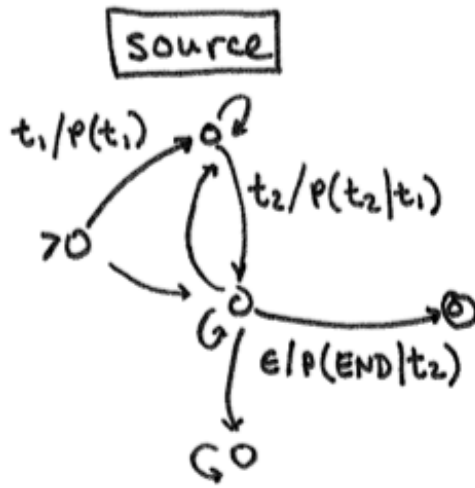
- use tag bigram as a language model
- channel model is context-indep.



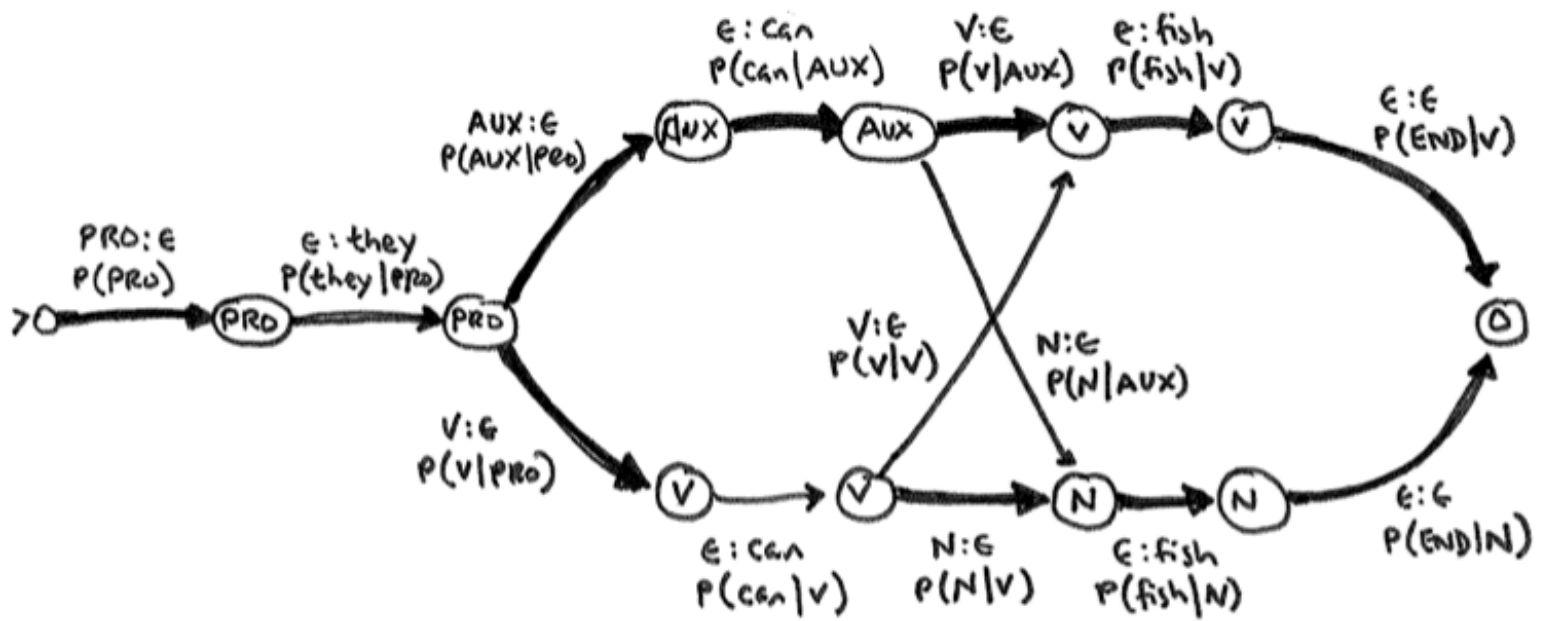
# Work out the compositions

- if you want to implement Viterbi...
- case 1: language model is a tag unigram model
  - $p(t_1 \dots t_n) = p(t_1)p(t_2) \dots p(t_n)$
  - how many states do you get?
- case 2: language model is a tag bigram model
  - $p(t_1 \dots t_n) = p(t_1)p(t_2 | t_1) \dots p(t_n | t_{n-1})$
  - how many states do you get?
- case 3: language model is a tag trigram model...

# The case of bigram model

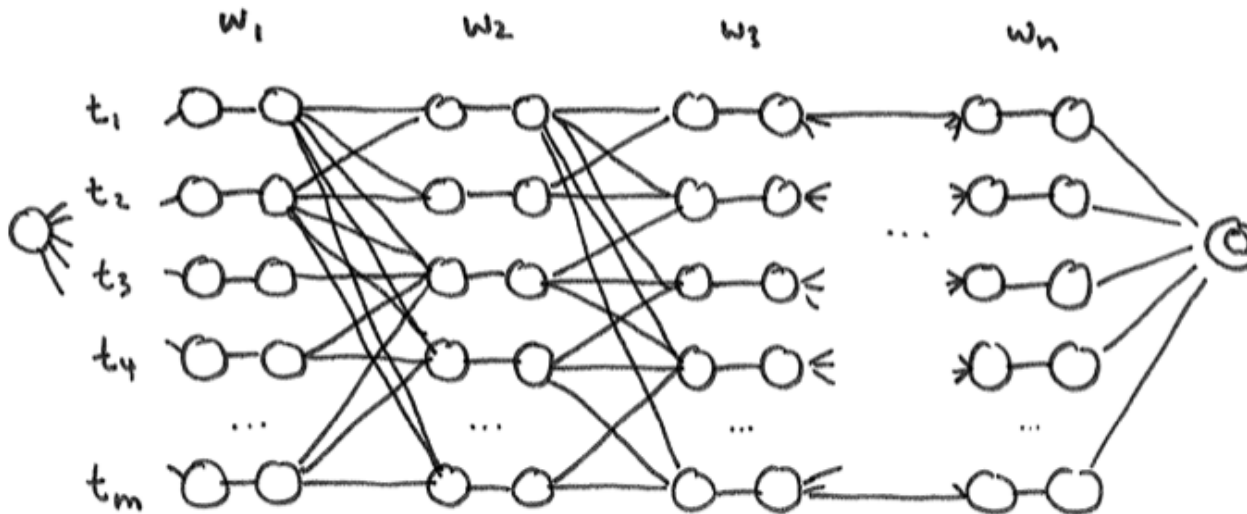


context-dependence (from LM)  
propagates left and right!

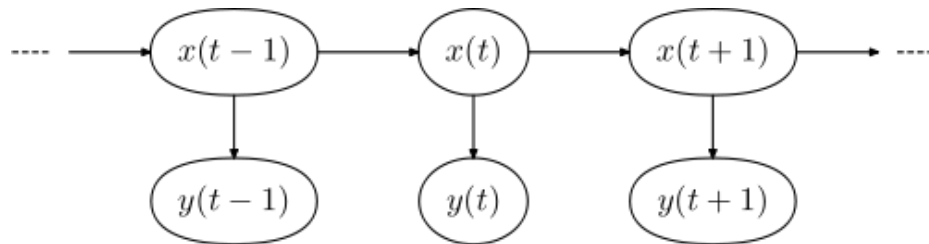


# In general...

- bigram LM with context-independent CM
  - $O(n m)$  states after composition
- g-gram LM with context-independent CM
  - $O(n m^{g-1})$  states after composition
  - the g-gram LM itself has  $O(m^{g-1})$  states



# HMM Representation



- HMM representation is not explicit about the search
  - “hidden states” have choices over “variables”
  - in FST composition, paths/states are explicitly drawn

# Viterbi for argmax

Viterbi search for  $\underset{t \dots t}{\operatorname{argmax}} P(t \dots t) \cdot P(w \dots w | t \dots t)$  :

```
for j = 1 to m
  Q[1,j] = P(tj) · P(w1 | tj)
```

```
for i = 2 to n
  for j = 1 to m
    Q[i,j] = 0
    best-prev[i,j] = 0
    best-score = -∞
    for k = 1 to m
      r = P(tj | tk) · P(wi | tj) · Q[i-1,k]
      if r > best-score
        best-score = r
        best-prev[i,j] = k
    Q[i,j] = r
```

$Q[i,j]$  = cost of shortest path ending with word  $i$  getting assigned tag  $j$ .

sets back pointers

```
final-best = 0
final-score = -∞
for j = 1 to m
  if Q[n,j] > final-score
    final-score = Q[n,j]
    final-best = j
```

```
print tfinal-best
current = final-best
for i = n-1 down to 1
  current = best-prev[i+1, current]
  print tcurrent
```

prints best tags in reverse order

how about unigram?

# Viterbi Tagging Example

given

Prab of	START	PRO	V	N	AUX
END		.1	.1	.1	.1
PRO	.6				
V	.05	.6		.2	.9
N	.3		.9	.7	
AUX	.05				

given

Prab of	PRO	V	N	AUX
they	.07			
can		$10^{-5}$	$10^{-4}$	.21
fish		$10^{-4}$	$10^{-4}$	

Q1. why is this table not normalized?

Q2. is "fish" equally likely to be a V or N?

Q3: how to train  $p(w|t)$ ?

	they	can	fish
PRO	$Q = P(\text{PRO} \text{START}) \cdot P(\text{they} \text{PRO})$ $= .6 \cdot .07 = .042$	$Q = 0$ $P(\text{can} \text{PRO}) = 0$	$Q = 0$
V	$Q = 0$ $P(\text{they} V) = 0$	$Q = \max \langle .042 \cdot .6 \cdot 10^{-5} \rangle$ $= .00000252$ $bp = \text{PRO}$	$Q = \max \langle .00000252 \cdot 0 \cdot 10^{-4}, .002646 \cdot .9 \cdot 10^{-4} \rangle$ $= .0000023814$ $bp = \text{AUX}$
N	$Q = 0$	$Q = 0$ $P(N \text{PRO}) = 0$	$Q = \max \langle .00000252 \cdot .9 \cdot 10^{-4}, .002646 \cdot 0 \cdot 10^{-4} \rangle$ $= .0000000002268$ $bp = V$
AUX	$Q = 0$	$Q = \max \langle .042 \cdot .3 \cdot .21 \rangle$ $= .002646$ $bp = \text{PRO}$	$Q = 0$

$$Q[1,j] = P(t_j|\text{START}) \cdot P(w_1|t_j)$$

$$Q[i,j] = \max_k Q[i-1,k] \cdot P(t_j|t_k) \cdot P(w_i|t_j)$$

# A Side Note on Normalization

**NOTE**

final-best gives  $P(t\dots t) \cdot P(w\dots w | t\dots t)$

but this is not the same as  $P(t\dots t | w\dots w)$

e.g. suppose there is only one  $t\dots t$  (all words unambiguous)

then  $P(t\dots t | w\dots w) = 1$

need to divide

$$P(t\dots t | w\dots w) = \frac{P(t\dots t) \cdot P(w\dots w | t\dots t)}{P(w\dots w)} = \frac{P(t\dots t) \cdot P(w\dots w | t\dots t)}{\sum_{t\dots t} P(t\dots t) \cdot P(w\dots w | t\dots t)}$$

how to compute the  
normalization factor?

# Forward (sum instead of max)

Forward search:  $\sum_t P(t) \cdot P(w|t) = P(w)$

$$\alpha[1, j] = P(t_j | \text{START}) \cdot P(w_1 | t_j)$$

$$\alpha[i, j] = \sum_k \alpha[i-1, k] \cdot P(t_j | t_k) \cdot P(w_i | t_j)$$

no back pointer

$$P(w) = \sum_k \alpha[n, k]$$

"Forward" procedure for  $P(w \dots w)$

for  $j = 1$  to  $m$   
 $\alpha[1, j] = P(t_j) \cdot P(w_1 | t_j)$

for  $i = 2$  to  $n$   
for  $j = 1$  to  $m$   
 $\alpha[i, j] = 0$

for  $k = 1$  to  $m$   
 $\alpha[i, j] += P(t_j | t_k) \cdot P(w_i | t_j) \cdot \alpha[i-1, k]$

$P(w \dots w) = 0$

for  $j = 1$  to  $m$   
 $P(w \dots w) += \alpha[n, j]$

$\alpha[i, j]$  = costs of all paths ending w/ word  $w_i$  getting  $t_k$   $t_j$  (costs summed)

# Forward vs. Argmax

- same complexity, different semirings  $(+, \times)$  vs  $(\max, \times)$
- for g-gram LM with context-indep. CM
- time complexity  $O(n m^g)$  space complexity  $O(n m^{g-1})$

```
for j = 1 to m  
  Q[1, j] = ...
```

```
for j = 1 to m  
  for j2 = 1 to m  
    Q[2, j, j2] = ...
```

```
for i = 3 to n  
  for j = 1 to m  
    for j2 = 1 to m  
      Q[i, j, j2] = 0  
      best-pred[i, j, j2] = 0  
      best-score = -∞  
      for k = 1 to m  
        r = P(tj2 | tj) · P(wi | tj2) · Q[i-1, k, j]  
        if r > best-score ...
```

$O(nm^3)$  complexity

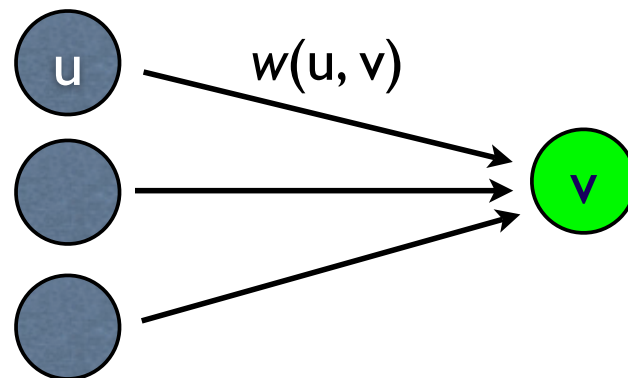
# Viterbi for DAGs with Semiring

1. topological sort

$$(A, \oplus, \otimes, \bar{0}, \bar{1})$$

2. visit each vertex  $v$  in sorted order and do updates

- for each **incoming** edge  $(u, v)$  in  $E$
- use  $d(u)$  to update  $d(v)$ :  $d(v) \oplus = d(u) \otimes w(u, v)$
- key observation:  $d(u)$  is fixed to optimal at this time

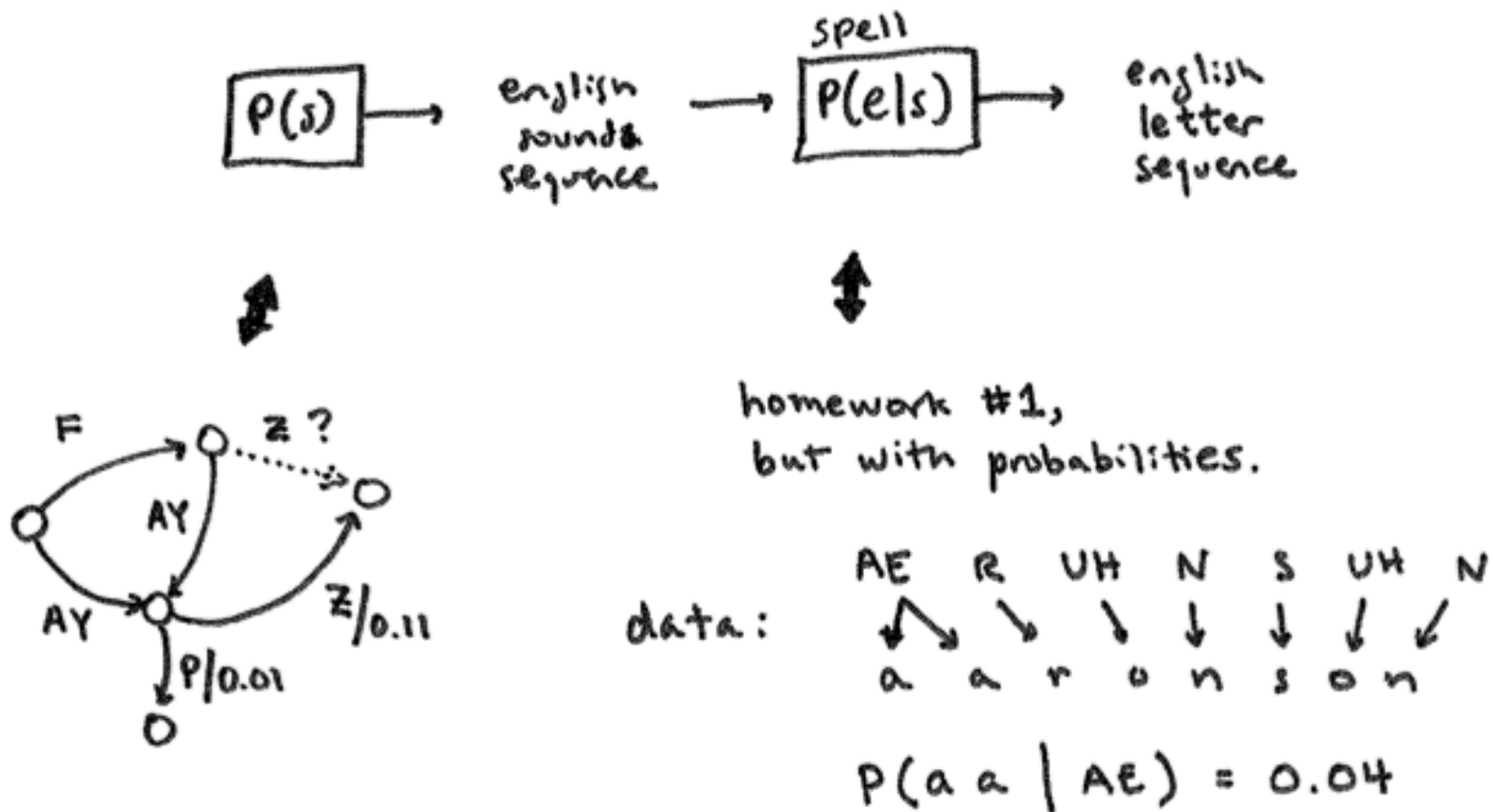


see tutorial on DP  
from course page

- time complexity:  $O(V + E)$

# Example: Pronunciation

- from spelling to sound



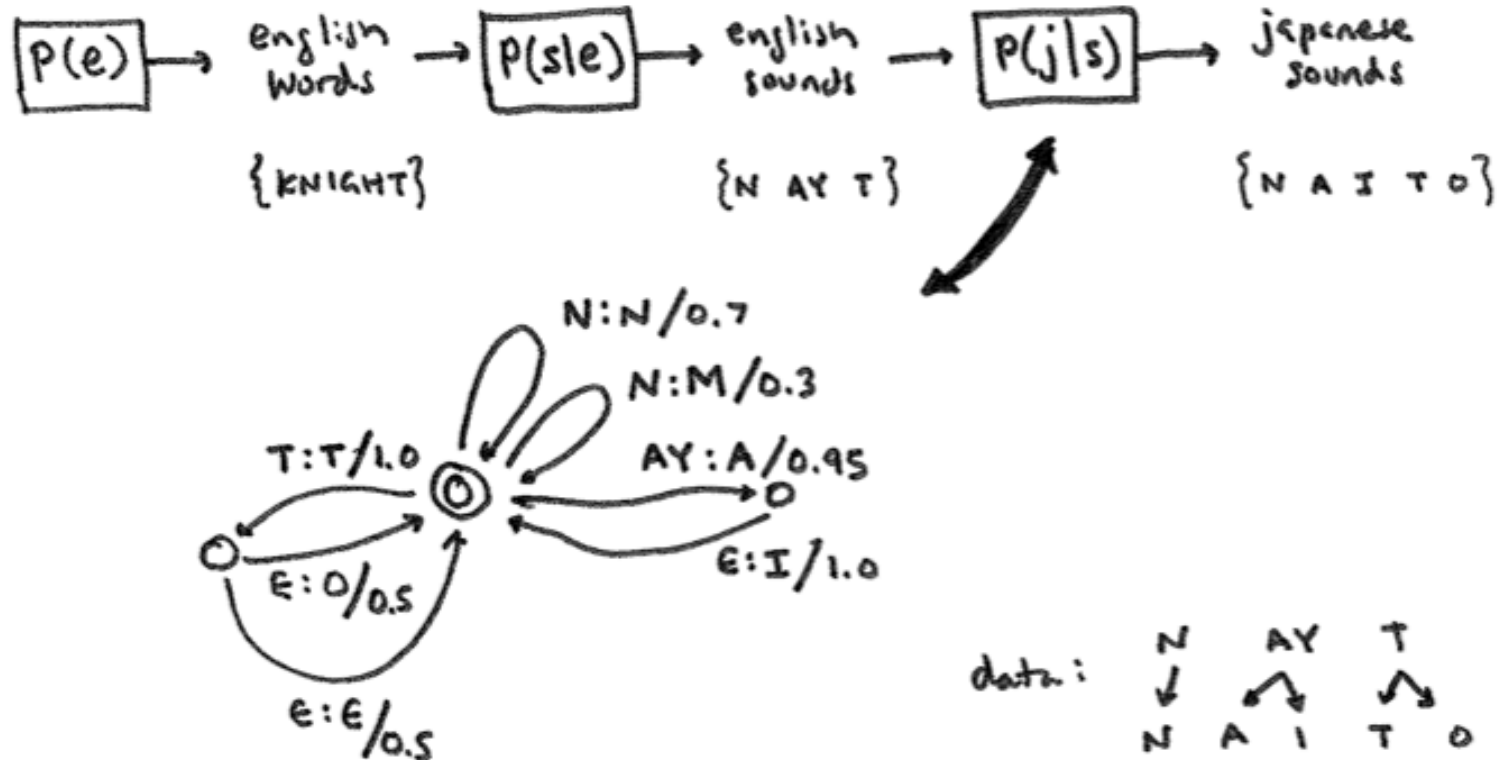
# Pronunciation Dictionary

- (hw3: eword-epron.data)
  - ...
  - AARON            EH R AH N
  - AARONSON    AA R AH N SAH N
  - ...
  - PEOPLE           P IY PAH L
  - VIDEO            V IH D IY OW
- you can train  $p(s..s|w)$  from this, but what about unseen words?
- also need alignment to train the channel model  $p(s|e)$  &  $p(e|s)$

# From Sound to Spelling

- input: HH EH L OW B EH R
- output: H E L L O B E A R or H E L O B A R E ?
- $p(e) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(s|w) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s \Rightarrow p(s)$
- $p(w) \Leftarrow w \Leftarrow p(w|e) \Leftarrow e \Leftarrow p(e|s) \Leftarrow s \Leftarrow p(s)$
- $w \Leftarrow p(w|s) \Leftarrow s \Leftarrow p(s)$
- can you further improve from these?

# Example: Transliteration



- KEVIN KNIGHT  $\Rightarrow$  KH EHVH IH N NAY T  
KEBIN NAITO

# Japanese 101 (writing systems)

- Japanese writing system has four components
  - Kanji (Chinese chars): nouns, verb/adj stems, CJKV names
    - 日本 “Japan” 東京 “Tokyo” 電車 “train” 食べる “eat [inf.]”
  - Syllabaries
    - Hiragana: function words (e.g. particles), suffices
      - で de (“at”) か ka (question) 食べました “ate”
    - Katakana: transliterated foreign words/names
      - コーヒー koohee (“coffee”)
  - Romaji (Latin alphabet): auxiliary purposes

# Why Japanese uses Syllabries

- all syllables are:  
[consonant] + vowel +  
[nasal *n*]
- 10 consonants, 5 vowels =  
50 basic syllables
  - plus some variations
- Other languages have way  
more syllables, so they do  
*alphabets*
- read the Writing Systems  
tutorial from course page!

あ ア a	い イ i	う ウ u	え エ e	お オ o
か カ ka	き キ ki	く ク ku	け ケ ke	こ コ ko
さ サ sa	し シ shi	す ス su	せ セ se	そ ソ so
た タ ta	ち チ chi	つ ツ tsu	て テ te	と ト to
な ナ na	に ニ ni	ぬ ヌ nu	ね ネ ne	の ノ no
は ハ ha	ひ ヒ hi	ふ フ hu / fu	へ ヘ he	ほ ホ ho
ま マ ma	み ミ mi	む ム mu	め メ me	も モ mo
や ヤ ya		ゆ ユ yu		よ ヨ yo
ら ラ ra	り リ ri	る ル ru	れ レ re	ろ ロ ro
わ ワ wa	<a href="http://brng.jp/90459562">http://brng.jp/ 90459562</a>			を ヲ wo
ん ン n				

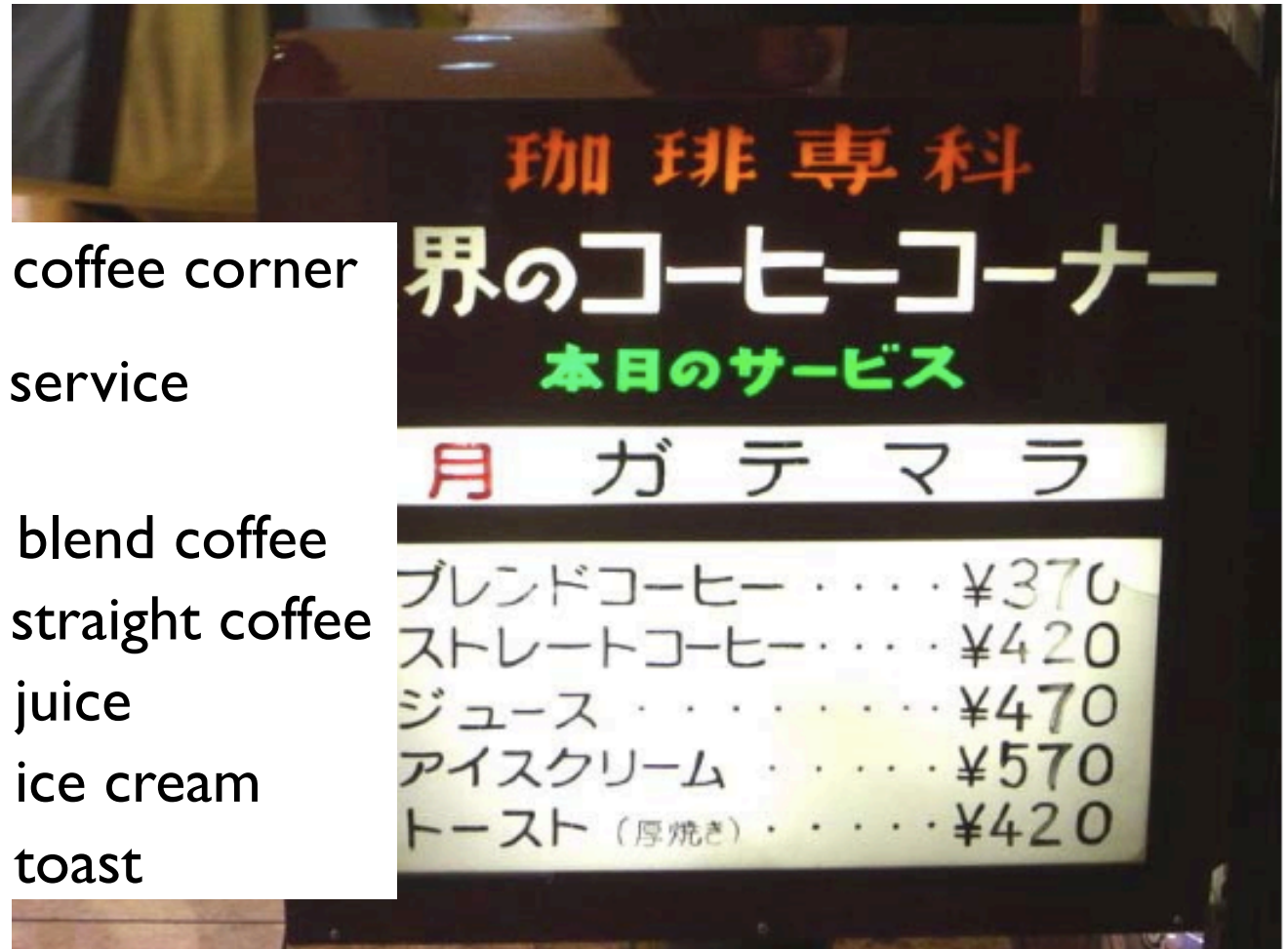
# Katakana Transliteration Examples

- コンピューター
- ko n py u - ta -
- kompyuuta (uu=û)
- computer
- アイスクリーム
- a i su ku ri - mu
- aisukuriimu
- ice cream
- アンドリュー・ビタビ
- ヨーグルト
- andoryuubitabi
- yo - gu ru to
- Andrew Viterbi
- yogurt

# Katakana on Streets of Tokyo

from Knight & Sproat 09

- koohiikoonaa coffee corner
- saabisu service
- bulendokoohii blend coffee
- sutoreetokoohii straight coffee
- juusu juice
- aisukuriimu ice cream
- toosuto toast



# Japanese $\Leftrightarrow$ English: Cascades

- your job in HW3: decode Japanese Katakana words (transcribed in Romaji) back to English words
  - koohiikoonaa  $\Rightarrow$  coffee corner
- what about duplicate paths with same string??
  - n-best crunching, or weighted determinization (see extra reading)

# Example: Word Segmentation

- you noticed that Japanese (e.g., Katakana) is written *without* spaces between words
  - in order to guess the English you also do segmentation
  - e.g. アイスクリーム: ice cream
- this is a more important issue in Chinese
  - 南京市长江大桥
- also in Korean, Thai, and other East Asian Languages
- also in English: sounds => words (speech recognition)

# Chinese Word Segmentation

民主

min-zhu

people-dominate

“democracy”



this was 5 years ago.

now Google is  
good at segmentation!

江泽民 主席

jiang-ze-min zhu-xi

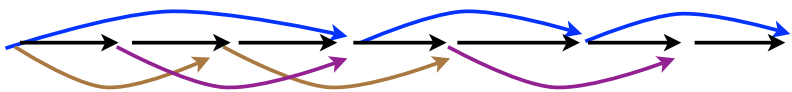
... - ... - people dominate-podium

“President Jiang Zemin”



下雨天 地面积水

xia yu tian di mian ji shui



graph search

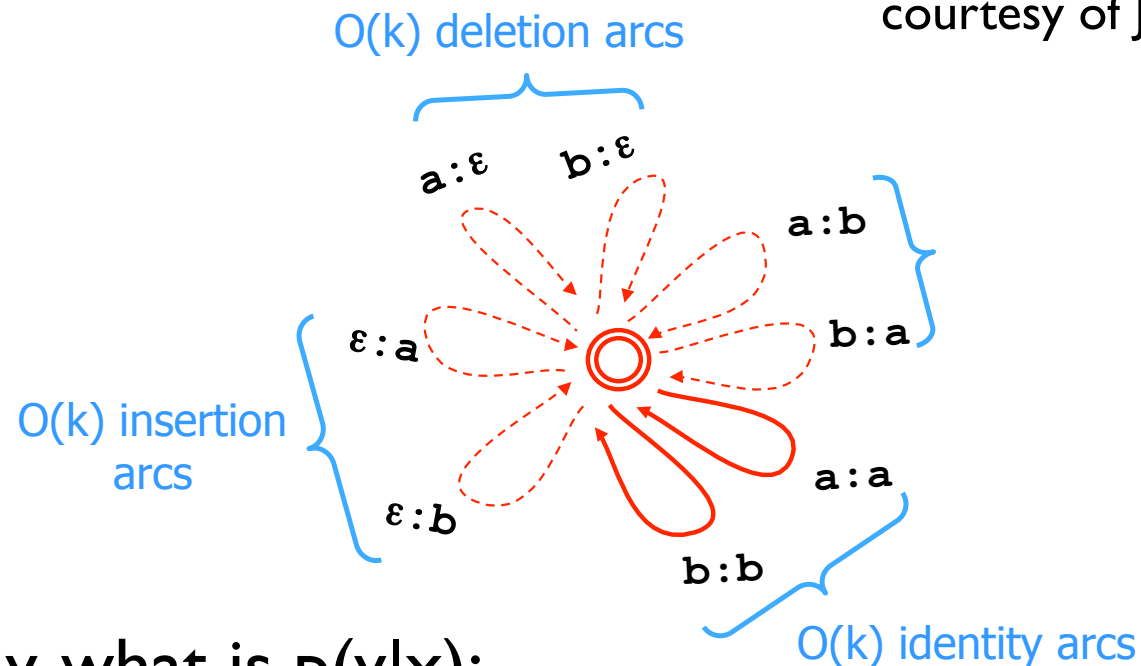
下雨天 地面积水

tagging problem

# Word Segmentation Cascades

# Example: Edit Distance

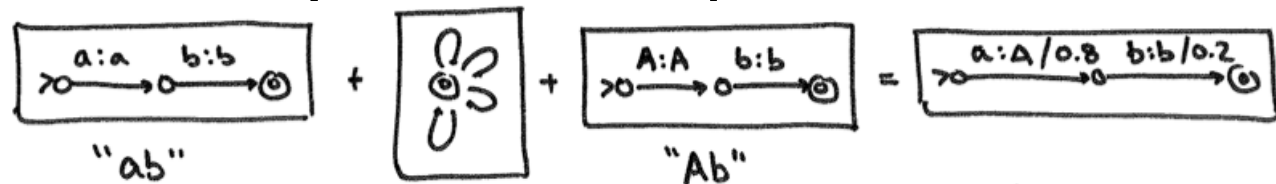
courtesy of Jason Eisner



- a) given  $x, y$ , what is  $p(y|x)$ ;
- b) what is the most likely seq. of operations?
- c) given  $x$ , what is the most likely output  $y$ ?
- d) given  $y$ , what is the most likely input  $x$  (with LM) ?

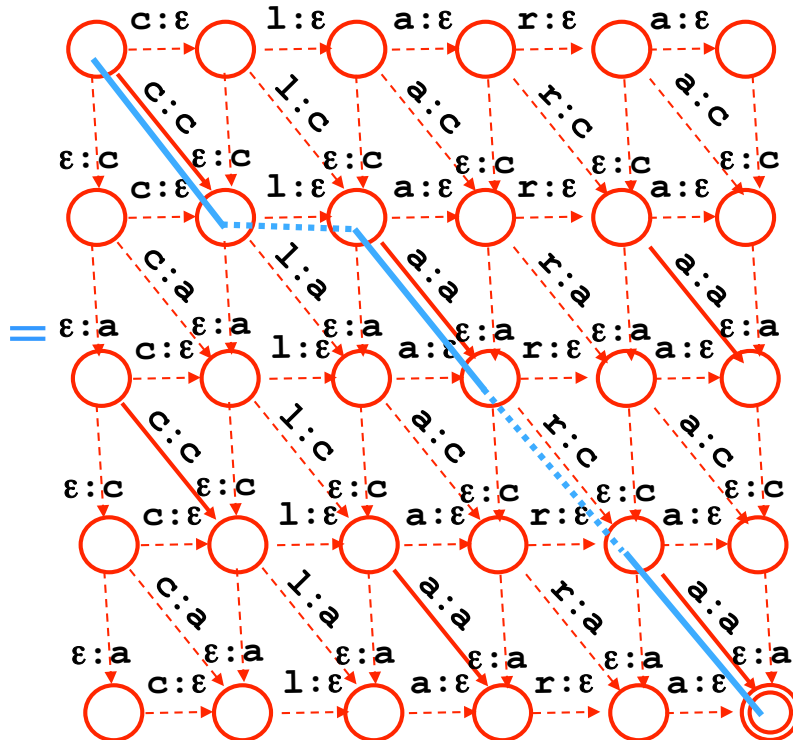
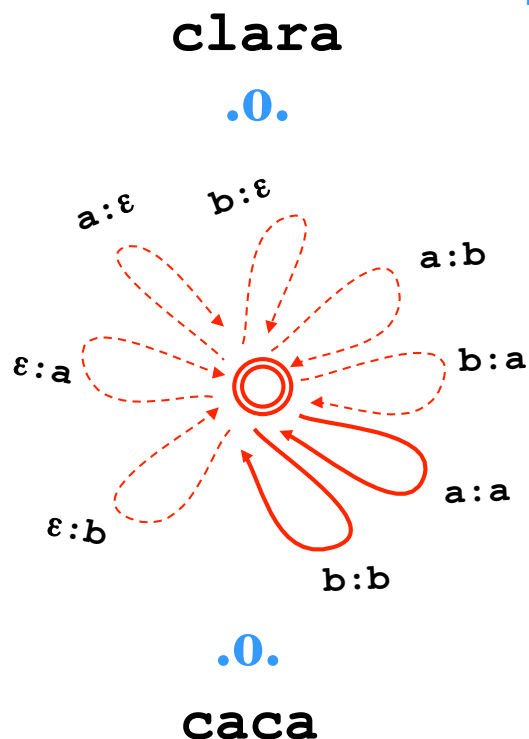
# Given x and y...

- given x, y a) what is  $p(y | x)$ ? (sum of all paths)
- b) what is the most likely conversion path?



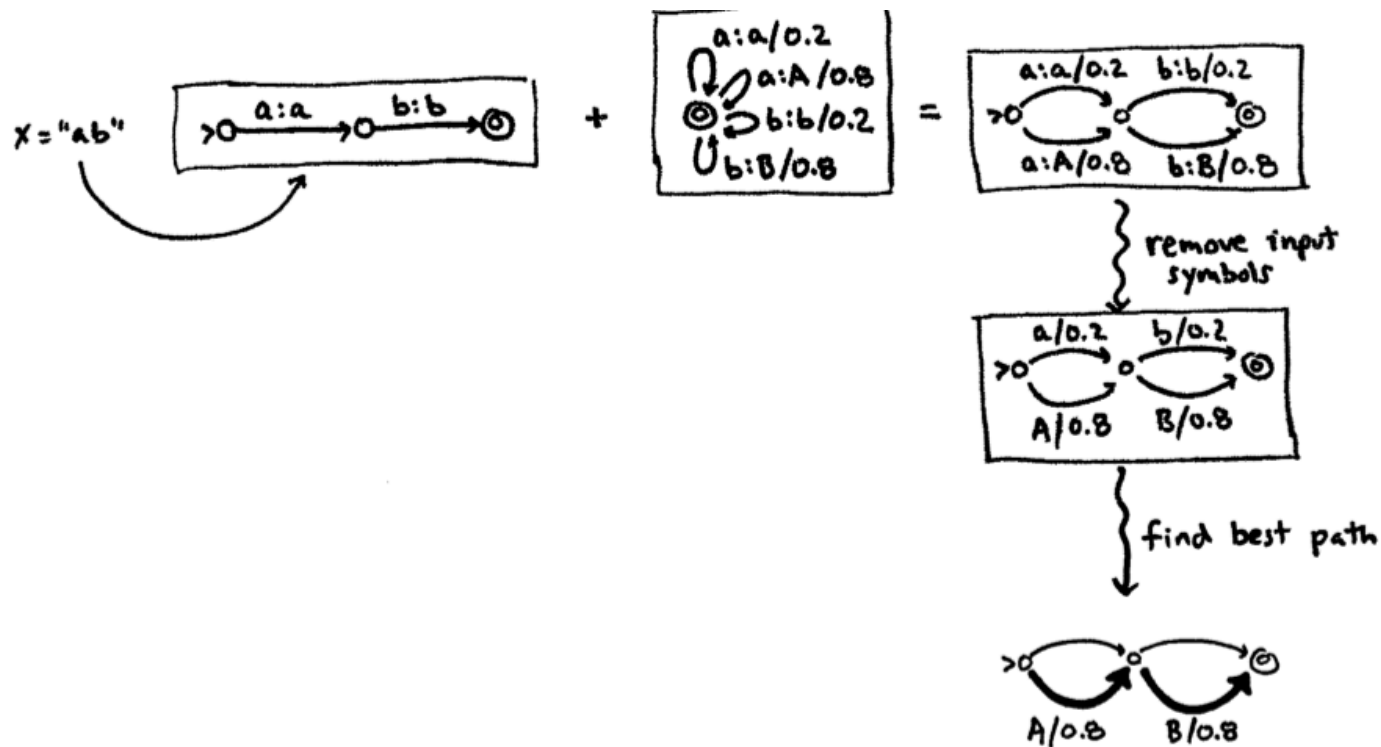
$$P(Ab|ab) = 0.16$$

Best path (by Dijkstra's algorithm)



# Most Likely “Corrupted Output”

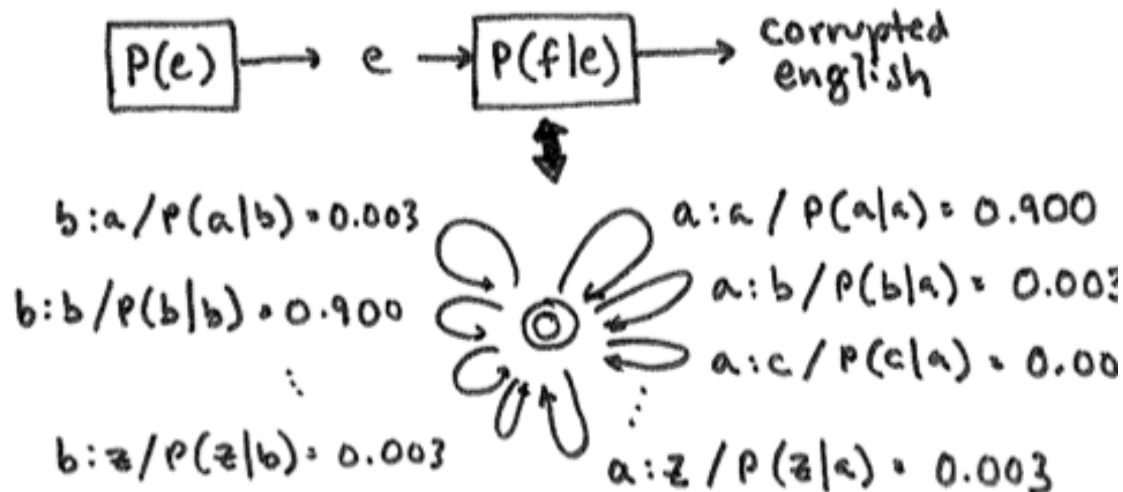
- c) given correct English  $x$ , what’s the corrupted  $y$  with the highest score?



$$\text{so, } \underset{y}{\operatorname{argmax}} P(y|ab) = AB$$

# DP for “most likely corrupted”

## d) Most Likely “Original Input”



- ignores insertions/deletions
- similar to "bad OCR" channel

- using an LM  $p(e)$  as source model for *spelling correction*
  - case 1: letter-based language model  $p_L(e)$
  - case 2: word-based language model  $p_w(e)$
- How would dynamic programming work for cases 1/2?

# Dynamic Programming for d)

# Summary of Edit Distance