

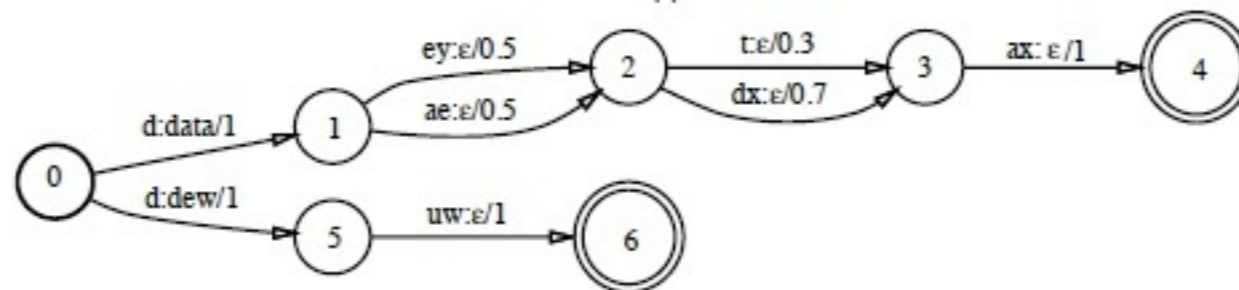
Language Technology

CUNY Graduate Center Spring 2013

Unit 1: Sequence Models

Lectures 7-8: Stochastic String Transformations

(a.k.a. “channel-models”)



Professor Liang Huang

liang.huang.sh@gmail.com

required

hard

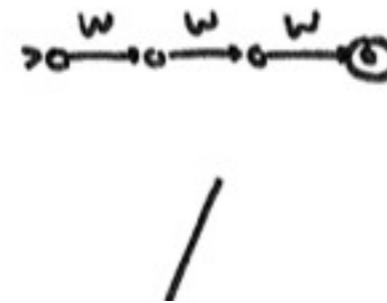
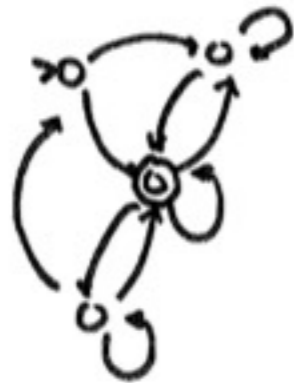
optional

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

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

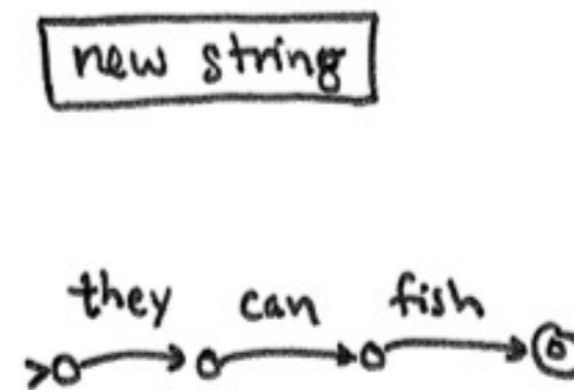
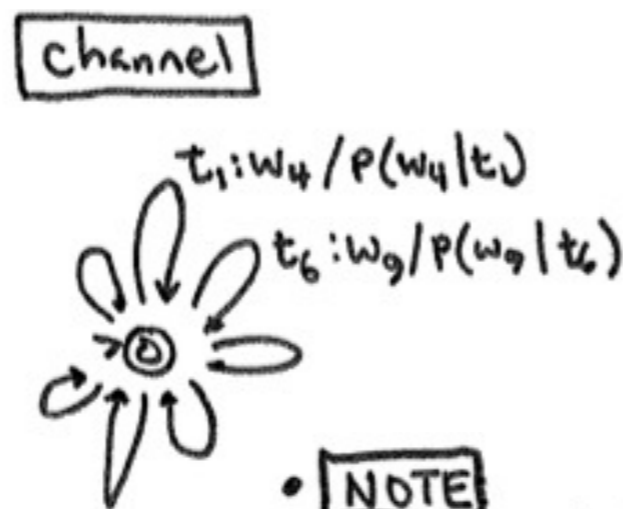
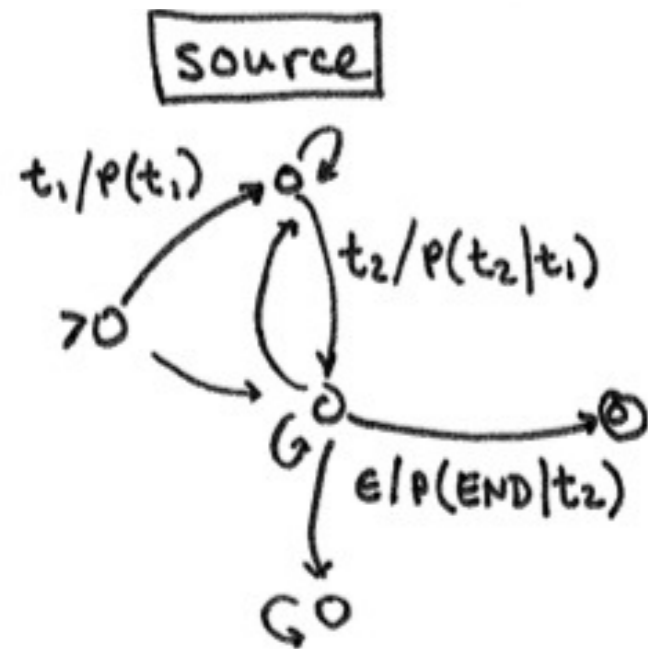
$$\sim P(t \dots t) \cdot P(w \dots w \mid t \dots t)$$

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

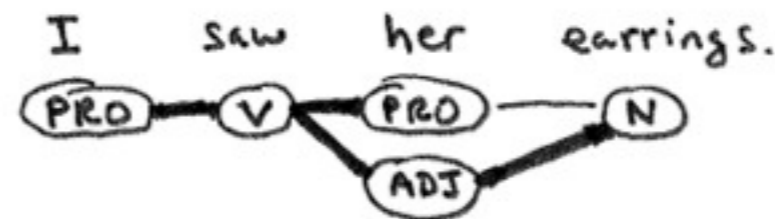
local grammar preference

lexical preference

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



• **NOTE** choice of t_{i+1} is influenced by both left & right context



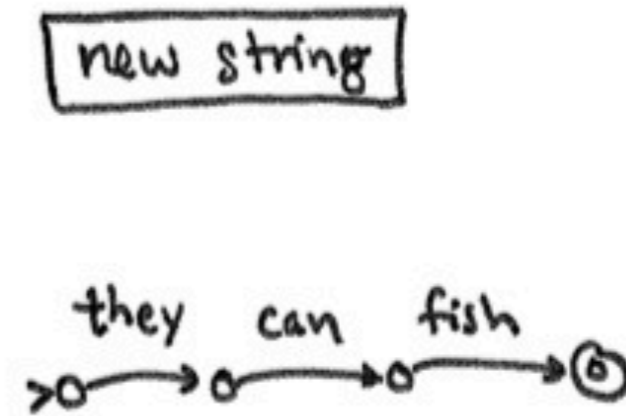
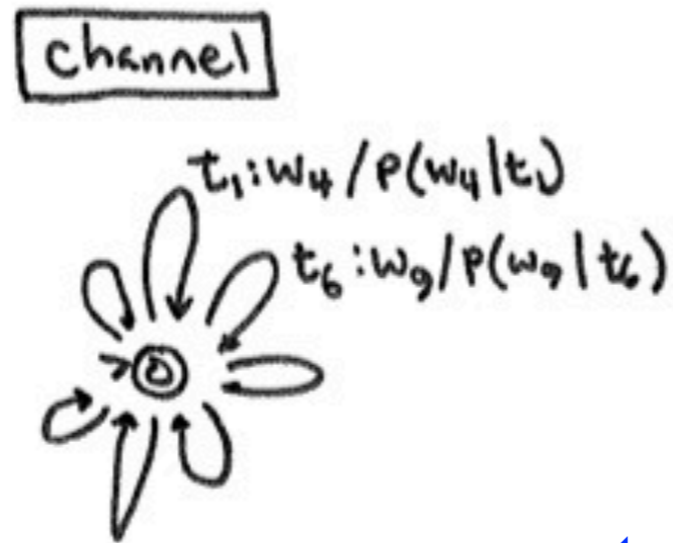
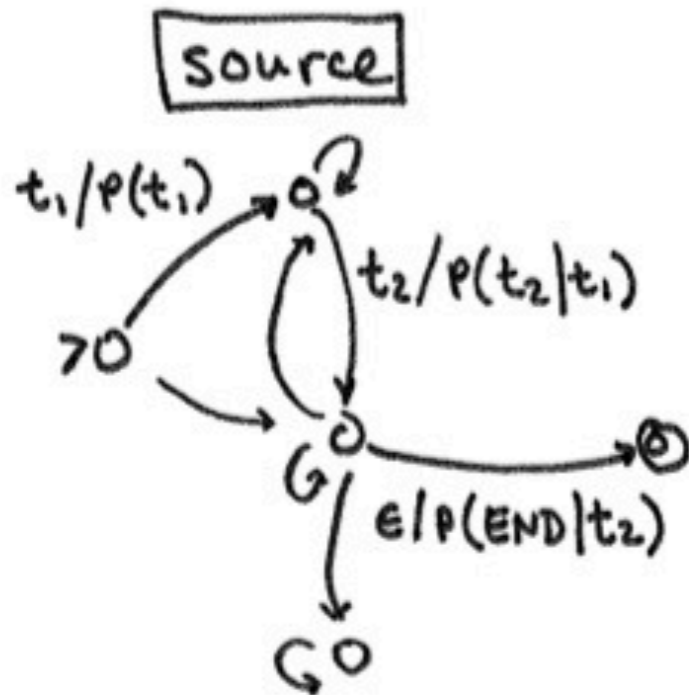
• **NOTE** influences can theoretically be long ranging



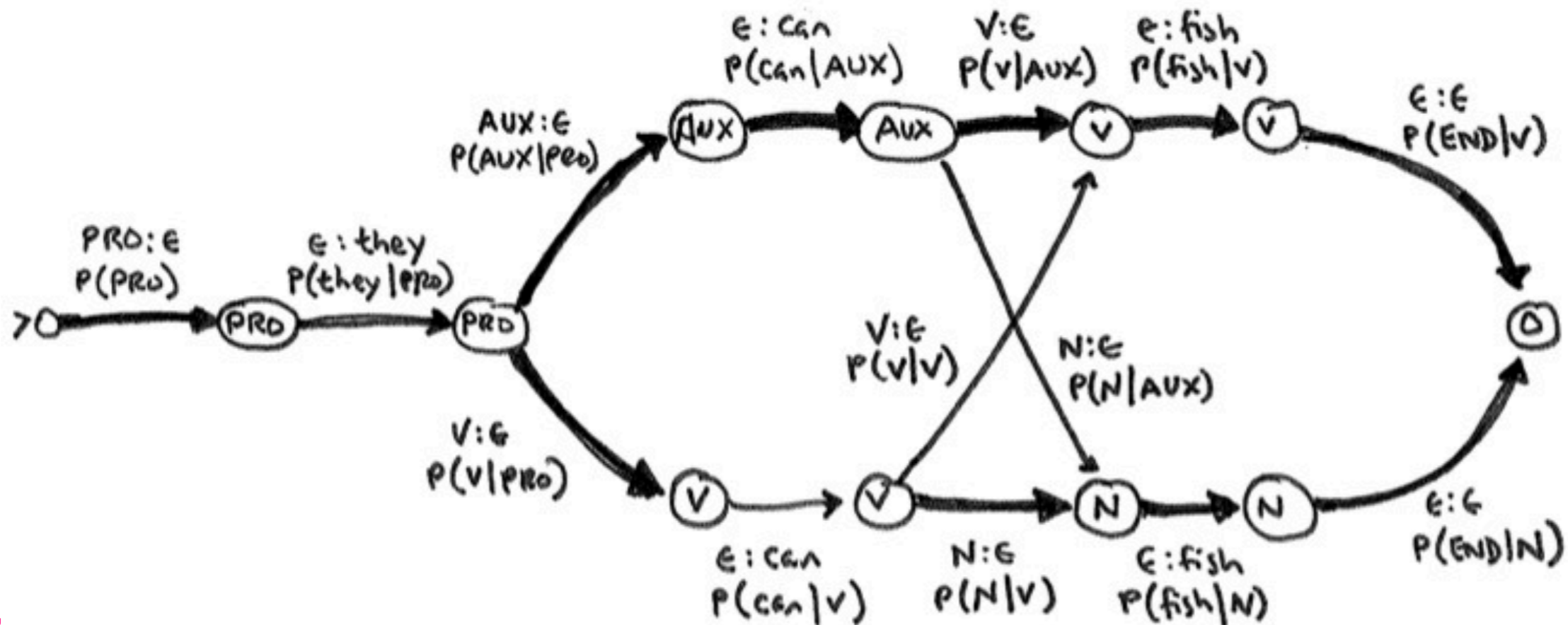
Work out the compositions

- 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

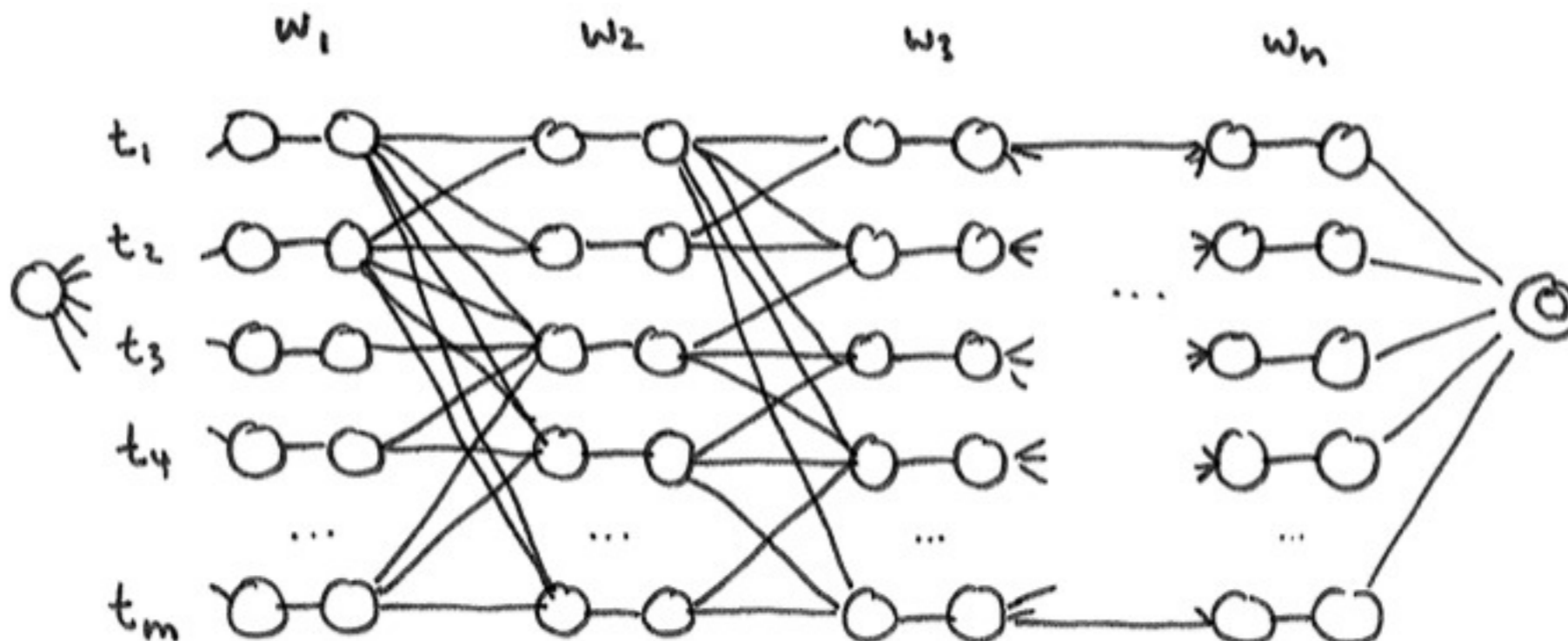


context-dependence (from LM)
 propagates left and right!

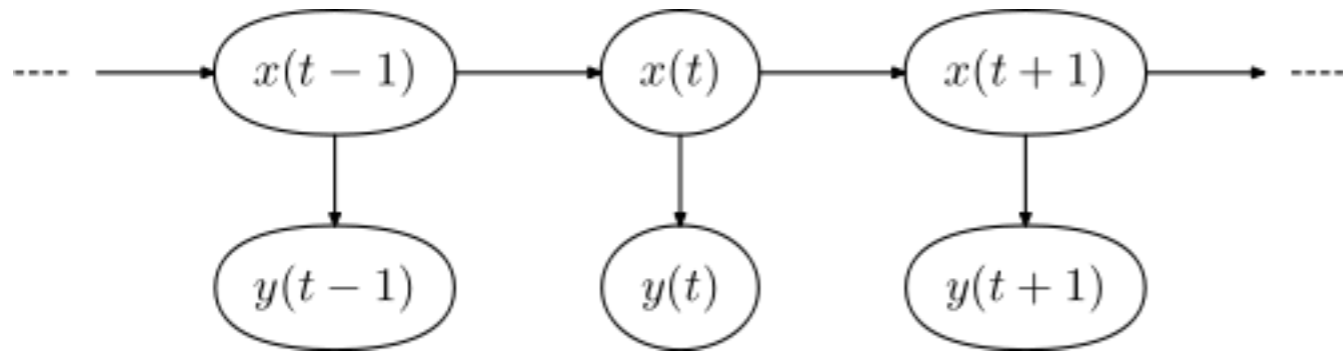


In general...

- 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 $\operatorname{argmax}_{t \dots t} P(t \dots t) \cdot P(w \dots w | t \dots t)$:

```
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?

Python implementation

Complete this Python code implementing the Viterbi algorithm for part-of-speech tagging. It should print a list of word/tag pairs, e.g. [('a', 'D'), ('can', 'N'), ('can', 'A'), ('can', 'V'), ('a', 'D'), ('can', 'N')].

```
1 from collections import defaultdict
2
3 best = defaultdict(lambda : defaultdict(float))
4 best[0][ "<s>" ] = 1
5 back = defaultdict(dict)
6
7 words = "<s> a can can can a can </s>".split()
8
9 tags = {"a": ["D"], "can": ["N", "A", "V"], "</s>": ["</s>"]} # possible tags for each word
10 ptag = {"D": {"N": 1}, "V": {"</s>": 0.5, "D": 0.5}, ... } # ptag[x][y] = p(y | x)
11 pword = {"D": {"a": 0.5}, "N": {"can": 0.1}, ... } # pword[x][w] = p(w | x)
12
13 for i, word in enumerate(words[1:], 1): # i=1,2,...; word=a,can,...
14     for tag in tags[word]:
15         for prev in best[i-1]:
16             if tag in ptag[prev]:
17                 score = best[i-1][prev] * ptag[prev][tag] * pword[tag][word]
18                 if score > best[i][tag]:
19                     best[i][tag] = score
20                     back[i][tag] = prev
21
22 def backtrack(i, tag):
23     if i == 0:
24         return []
25     return backtrack(i-1, back[i][tag]) + [(words[i], tag)]
26
27 print backtrack(len(words)-1, "</s>")[:-1]
```

Q: what about top-down recursive + memoization?

Viterbi Tagging Example

given

prob 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

prob 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	$Q = P(\text{PRO} \text{START}) \cdot P(\text{they} \text{PRO})$ $= .6 \cdot .07 = .042$	$Q = 0$ $P(\text{PRO} \text{PRO}) = 0$ $P(\text{can} \text{PRO}) = 0!$	$Q = 0$
V	$Q = 0$ $P(\text{they} \text{V}) = 0$	$Q = \max\langle .042 \cdot .6 \cdot 10^{-5} \rangle$ $= .000000252$ $bp = \text{PRO}$	$Q = \max\langle .000000252 \cdot 0 \cdot 10^{-4}, .002646 \cdot .9 \cdot 10^{-4} \rangle$ $= .00000023814$ $bp = \text{AUX}$
N	$Q = 0$	$Q = 0$ $P(\text{N} \text{PRO}) = 0$	$Q = \max\langle .000000252 \cdot .9 \cdot 10^{-4}, .002646 \cdot 0 \cdot 10^{-4} \rangle$ $= .00000000002268$ $bp = \text{V}$
AUX	$Q = 0$	$Q = \max\langle .042 \cdot .3 \cdot .21 \rangle$ $= .002646$ $bp = \text{PRO}$	$Q = 0$

bp = V

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

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

Excel Demo

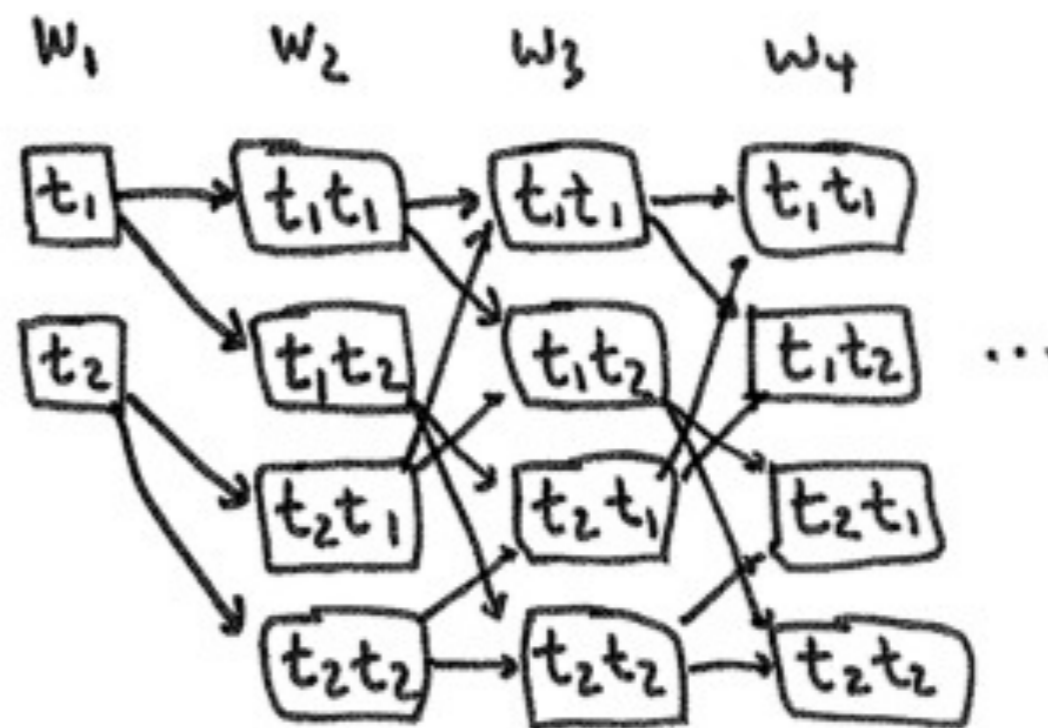
Trigram HMM

for $j = 1$ to m
 $Q_1[1, j] = \dots$

for $j = 1$ to m
 for $j_2 = 1$ to m
 $Q[2, j, j_2] = \dots$

for $i = 3$ to n
 for $j = 1$ to m
 for $j_2 = 1$ to m
 $Q[i, j, j_2] = 0$
 $\text{best-pred}[i, j, j_2] = 0$
 $\text{best-score} = -\infty$
 for $k = 1$ to m

$r = P(t_{j_2} | t_j) \cdot P(w_i | t_{j_2}) \cdot Q[i-1, k, j]$
 if $r > \text{best-score} \dots$



time complexity: $O(nT^3)$
 in general: $O(nT^g)$ for g -gram

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

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

$P(w \dots w) = 0$
for $j = 1$ to m
 $P(w \dots w) += \alpha[n, j]$

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 =  $-\infty$ 
```

```
      for k = 1 to m  
        r =  $P(t_{j2} | t_j) \cdot P(w_i | t_{j2}) \cdot Q[i-1, k, j]$   
        if r > best-score ...
```

$O(nm^3)$ complexity

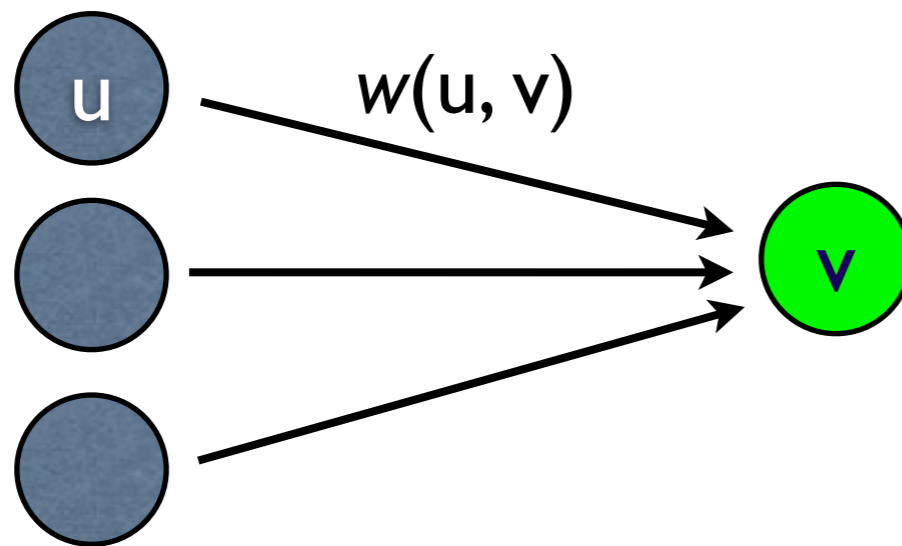
Viterbi for DAGs with Semiring

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

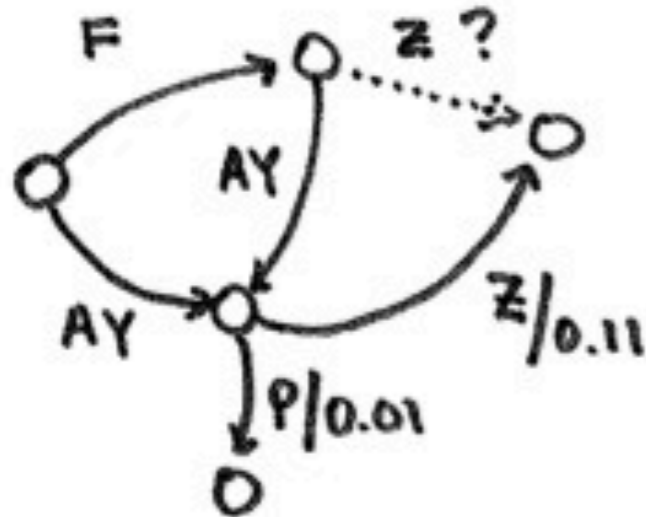
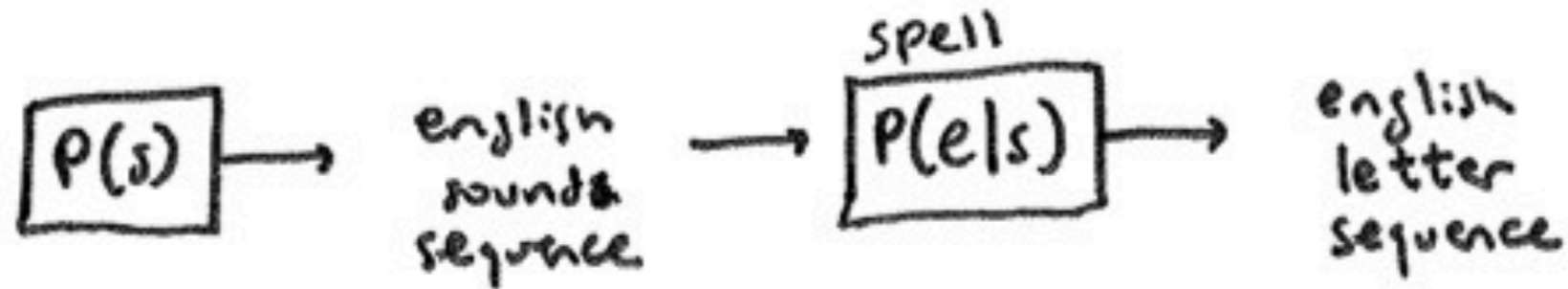


see tutorial on DP
from course page

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

(hw3) From Spelling to Sound

- word-based or char-based



homework #1,
but with probabilities.

data:

AE	R	UH	N	S	UH	N
↙ ↘	↘	↓	↓	↓	↓	↙
a	a	r	o	n	s	o

$$P(a a | AE) = 0.04$$

Pronunciation Dictionary

- (hw3: eword-epron.data) <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>
from CMU Pronunciation Dictionary
- ... 39 phonemes (15 vowels + 24 consonants)
- AARON EH RAH N
- AARONSON AA RAH N SAH N
- ...
echo 'W H A L E B O N E S' |
carmel -sriIEQk 5 epron.wfsa epron-espell.wfst
- 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 $p(s|e)$ & $p(e|s)$

CMU Dict: 39 Ame. Eng. Phonemes

WRONG! missing the SCHWA ə (merged with the STRUT ʌ “AH”)!

CMU/IPA	Example	Translation	CMU/IPA	Example	Translation
AA /ɑ/	o dd	AA D	K /k/	k ey	K IY
AE /æ/	a t	AE T	L /l/	l ee	L IY
AH /ʌ/	h u t	HH AH T	M /m/	m e	M IY
AO /ɔ:/	o ught	AO T	N /n/	k nee	N IY
AW /aʊ/	c ow	K AW	NG /ŋ/	ping ng	P IH NG
AY /aɪ/	h i de	HH AY D	OW /oʊ/	o at	OW T
B /b/	b e	B IY	OY /ɔɪ/	t oy	T OY
CH /tʃ/	ch eese	CH IY Z	P /p/	p ee	P IY
D /d/	d ee	D IY	R /r/	r ead	R IY D
DH /ð/	th ee	DH IY	S /s/	s ea	S IY
EH /ɛ/	E d	EH D	SH /ʃ/	sh e	SH IY
ER /ɜ/	h ur t	HH ER T	T /t/	t ea	T IY
EY /eɪ/	a te	EY T	TH /θ/	th eta	TH EY T AH
F /f/	f ee	F IY	UH /ʊ/	h ood	HH UH D
G /g/	g reen	G R IY N	UW /u/	t oo	T UW
HH /h/	h e	HH IY	V /v/	v ee	V IY
IH /ɪ/	i t	IH T	W /w/	w e	W IY
IY /i:/	e at	IY T	Y /j/	y ield	Y IY L D
JH /dʒ/	g ee	JH IY	Z /z/	z ee	Z IY
			ZH /ʒ/	u sual	Y UW ZH UW AH L

CMU Pronunciation Dictionary

WRONG! missing the SCHWA ə (merged with the STRUT ʌ “AH”)!
DOES NOT ANNOTATE STRESSES

A	AH
A	EY
AAA	T R IH P AH L EY
AABERG	AA B ER G
AACHEN	AA K AH N
...	
ABOUT	AH B AW T
...	
ABRAMOVITZ	AH B R AA M AH V IH T S
ABRAMOWICZ	AH B R AA M AH V IH CH
ABRAMOWITZ	AH B R AA M AH W IH T S
...	
FATHER	F AA DH ER
...	
ZYDECO	Z AY D EH K OW
ZYDECO	Z IH D AH K OW
ZYDECO	Z AY D AH K OW
...	
ZZZZ	Z IY Z

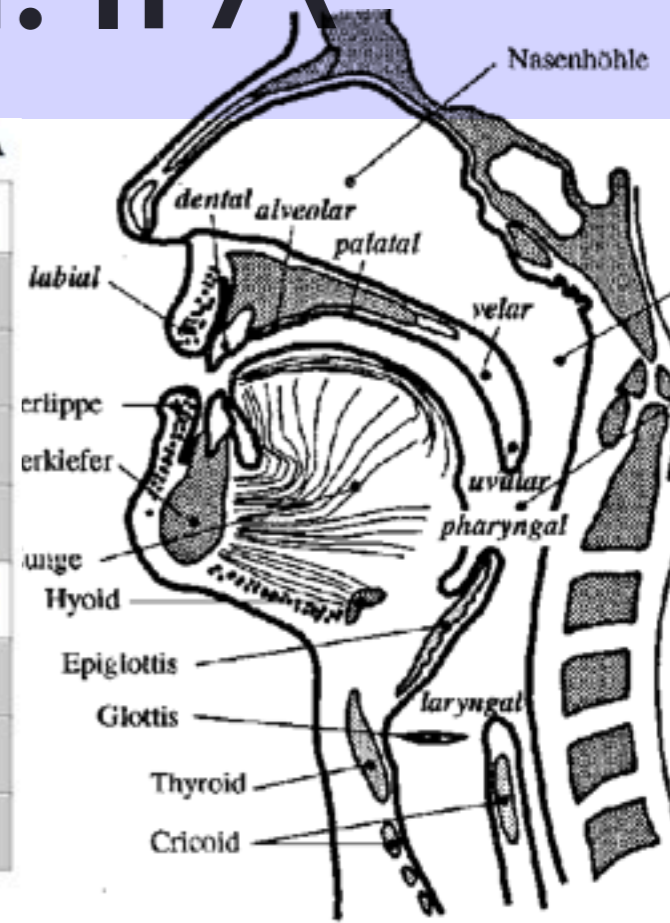
Linguistics Background: IPA

CONSONANTS (PULMONIC)

© 2005 IPA

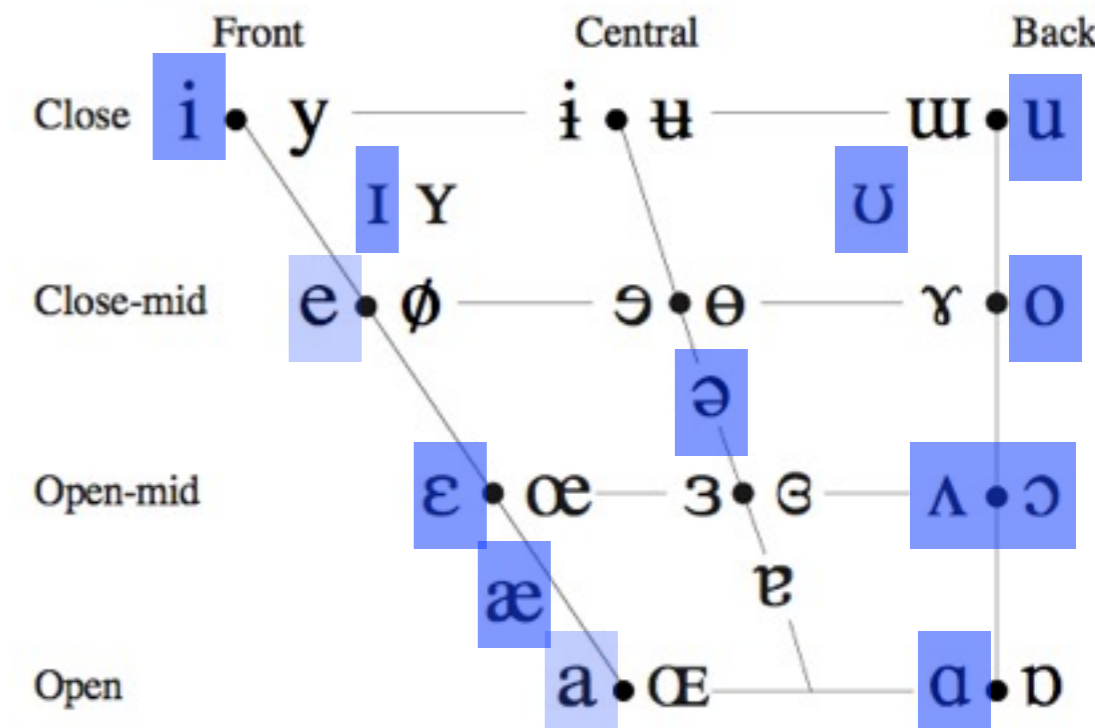
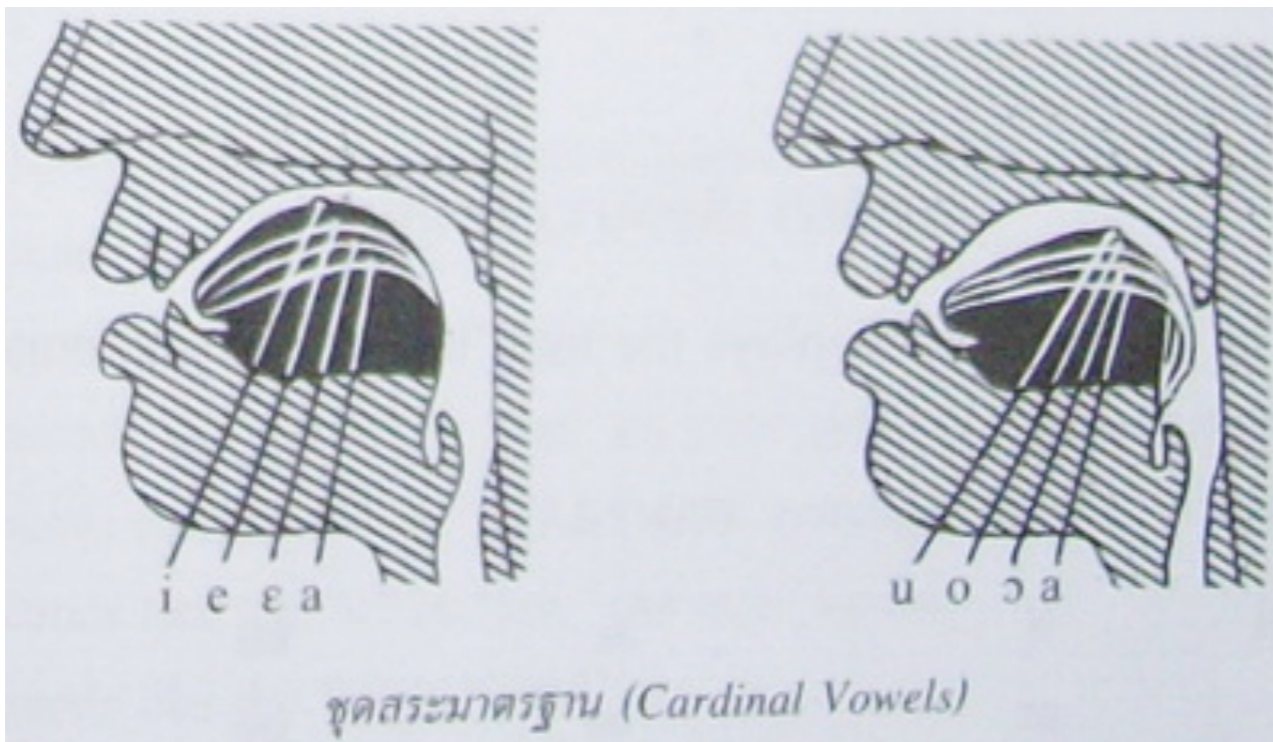
	Bilabial	Labiodental	Dental	Alveolar	Post alveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			ʀ					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.



Quellenangabe: aus [2] Seite 44 Abb. 17

VOWELS



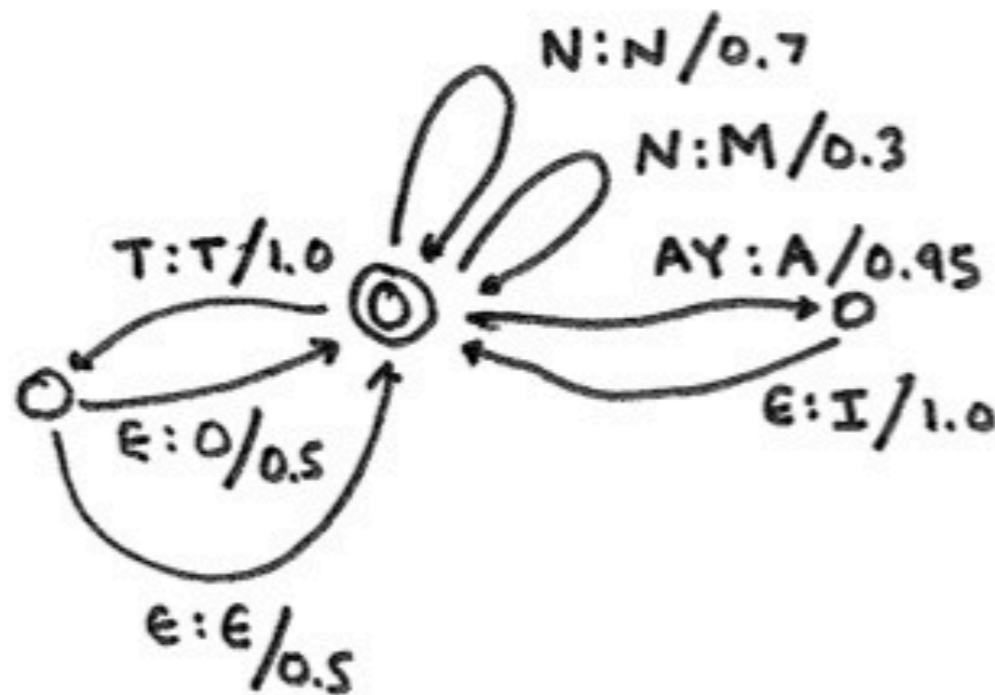
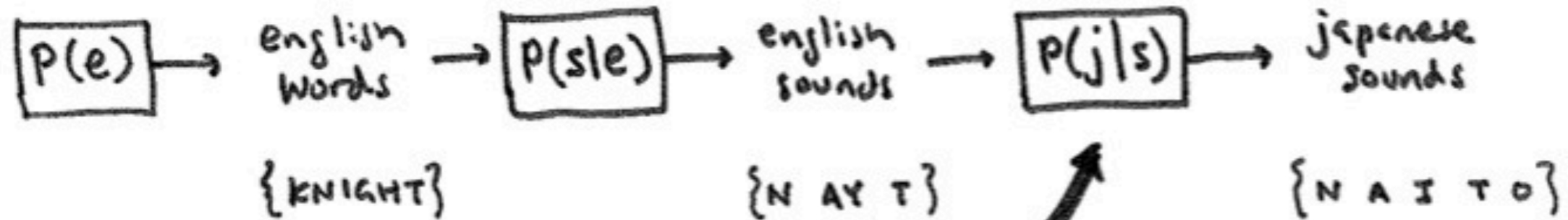
Where symbols appear in pairs, the one to the right represents a rounded vowel.

Linguistics Background: emic-etic

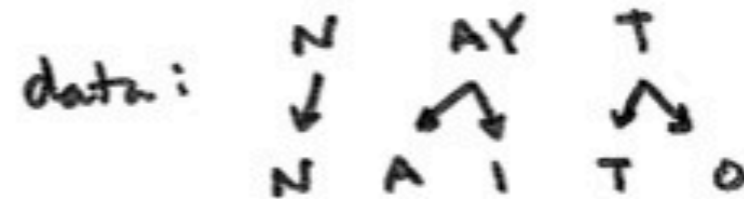
(hw3) From Sound to Spelling

- 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 ?
- $p(e) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(s|w) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s \Rightarrow p(s)$
- $p(w) \Leftarrow w \Leftarrow p(w|e) \Leftarrow e \Leftarrow p(e|s) \Leftarrow s \Leftarrow p(s)$
- $w \Leftarrow p(w|s) \Leftarrow s \Leftarrow p(s)$
- what else? `echo 'HH EH L OW' | carmel -sliOEQk 50 epron-espell.wfst
espell-eword.wfst eword.wfsa`

Example: Transliteration



- $V \Rightarrow B$: phoneme inventory mismatch
- $T \Rightarrow T$ O : phonotactic constraint

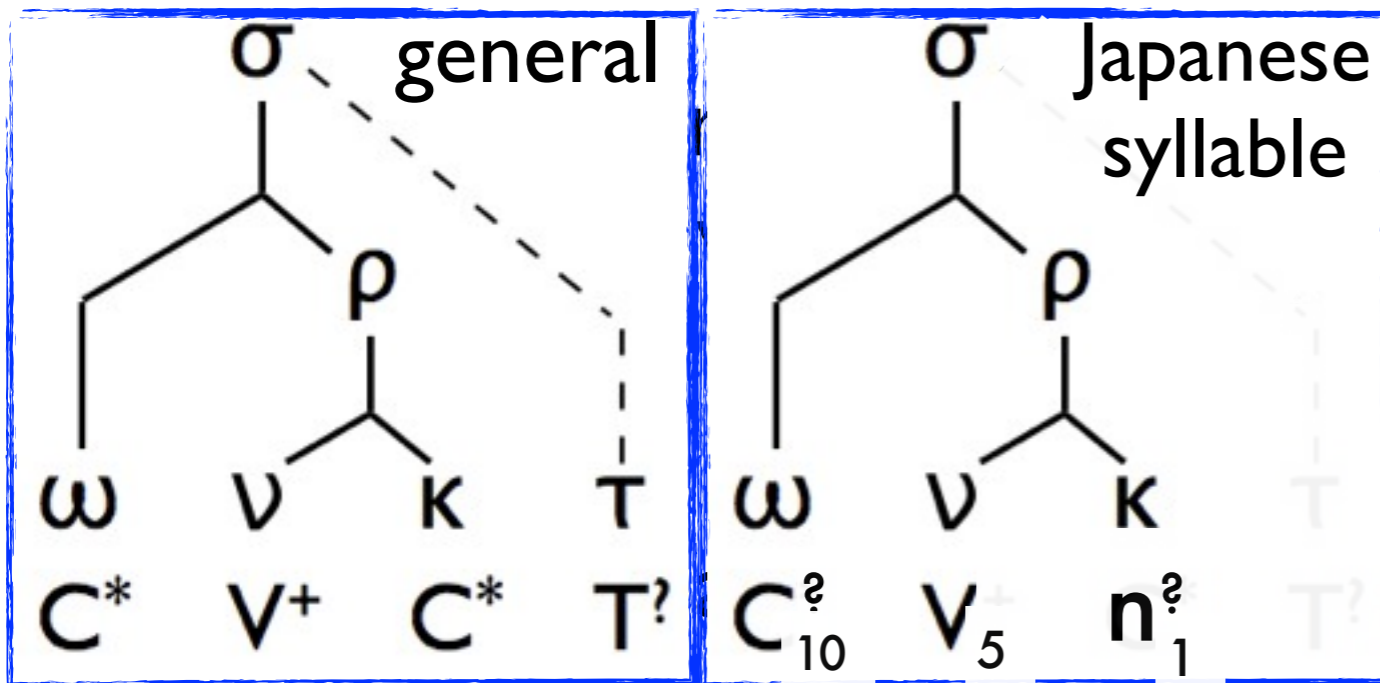


- KEVIN KNIGHT \Rightarrow KH EHVH IH N NAY T
 KE BIN NA I TO
 ケ ビン ナ イ ト

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 Syllabaries



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	http://brng.jp/ 90459562			を ヲ wo

- also possible for Mandarin
- other languages have many more syllables: use *alphabets*
- alphabet = 10+5; syllabary = 10x5
- read the Writing Systems tutorial from course page!

Japanese Phonemes (too few sounds!)

CONSONANTS (PULMONIC)

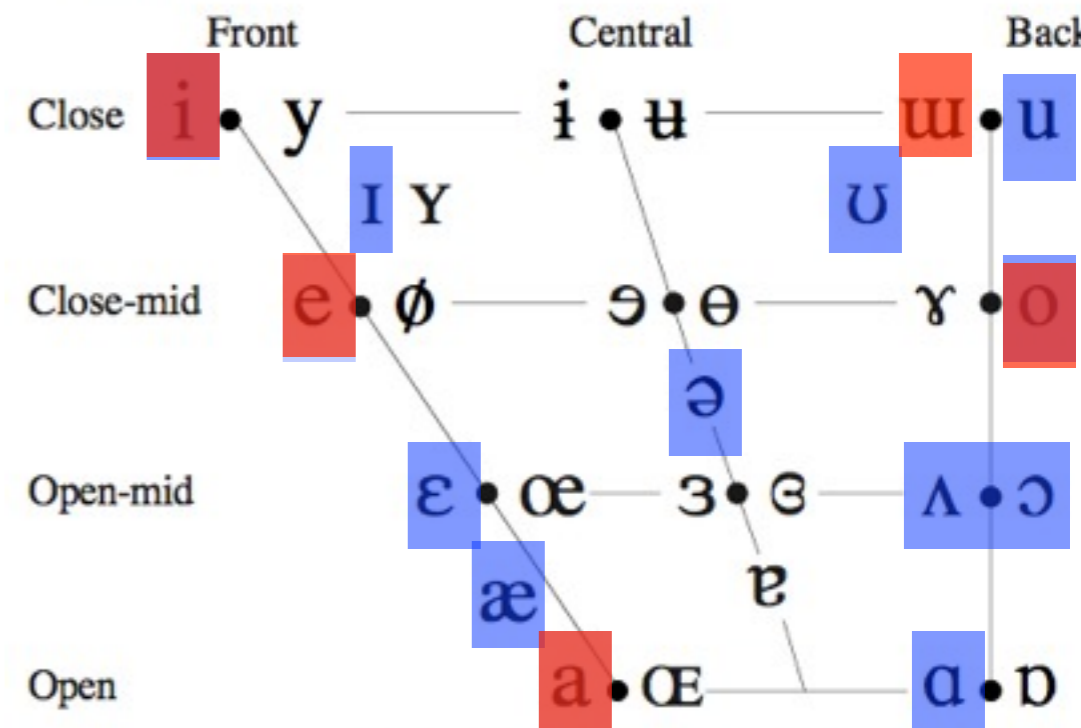
© 2005 IPA

	Bilabial	Labiodental	Dental	Alveolar	Post alveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			ʀ					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Eng
Jap

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

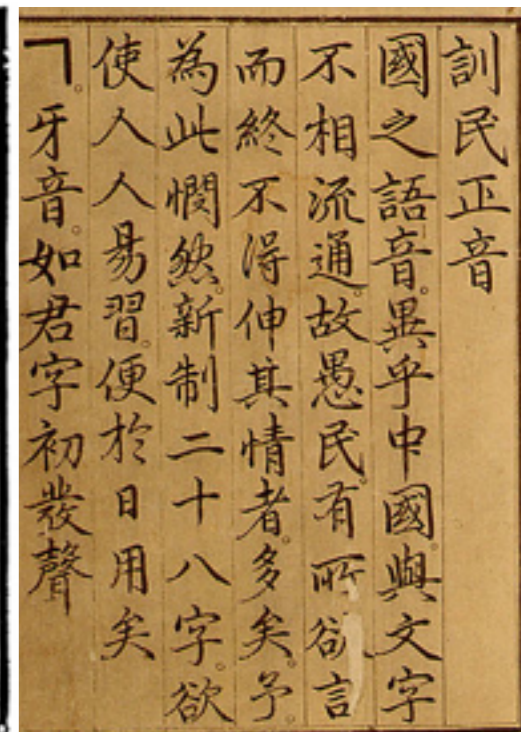
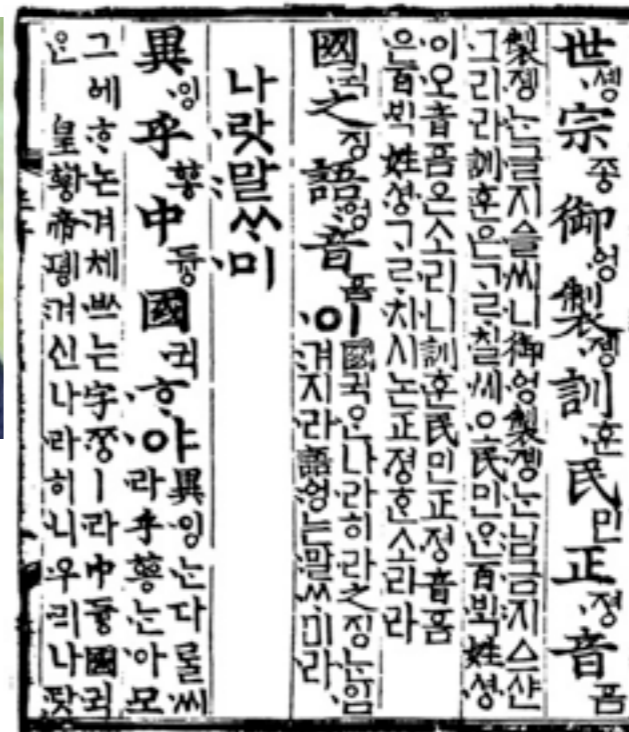
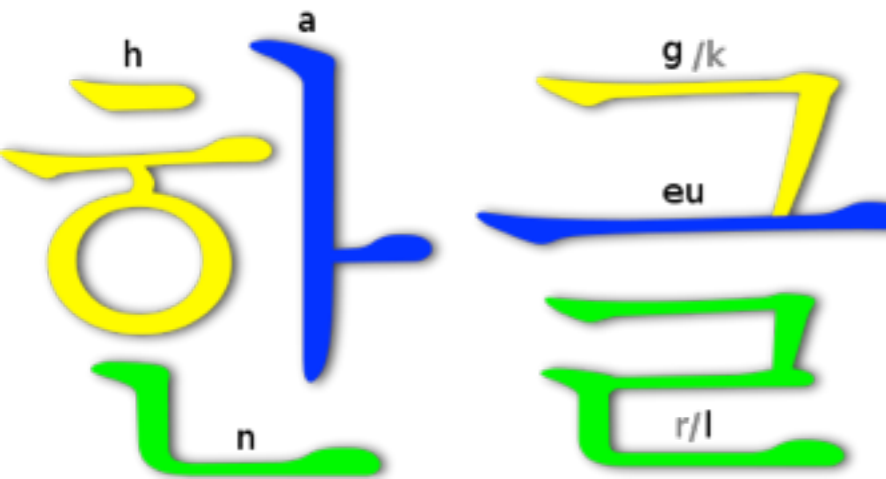
VOWELS



Where symbols appear in pairs, the one to the right represents a rounded vowel.

Aside: Is Korean a Syllabary?

- A: Hangeul is not a syllabary, but a “**featural alphabet**”
 - a special alphabet where shapes encode phonological features
- the inventor of Hangeul (c. 1440s) was the first real linguist



- 14 consonants: ㄱ *g*, ㄴ *n*, ㄷ *d*, ㄹ *l/r*, ㅁ *m*, ㅂ *b*, ㅅ *s*,
ㅇ *null/ng*, ㅈ *j*, ㅊ *ch*, ㅋ *k*, ㅌ *t*, ㅍ *p*, ㅎ *h*
- 5 double consonants: ㄲ *kk*, ㄸ *tt*, ㅃ *pp*, ㅆ *ss*, ㅉ *jj*
- 11 consonant clusters: ㄱㅅ *gs*, ㄴㅈ *nj*, ㄴㅎ *nh*, ㄹㅇ *lg*, ㄹㅁ *lm*, ㄹㅂ *lb*, ㄹㅅ *ls*, ㄹㅌ *lt*, ㄹㅍ *lp*, ㄹㅎ *lh*, ㅃㅅ *bs*
- 6 vowel letters: ㅏ *a*, ㅑ *eo*, ㅓ *o*, ㅕ *u*, ㅗ *eu*, ㅛ *i*
- 4 iotized vowels (with a *y*): ㅛ *ya*, ㅜ *yeo*, ㅠ *yo*, ㅠ *yu*
- 5 (iotized) diphthongs: ㅚ *ae*, ㅜ *yae*, ㅝ *e*, ㅞ *ye*, ㅟ *ui*
- 6 vowels and diphthongs with a *w*: ㅘ *wa*, ㅙ *wae*, ㅚ *oe*, ㅜ *wo*, ㅝ *we*, ㅟ *wi*

Q: 강남 스타일 = ?

Katakana Transliteration Examples

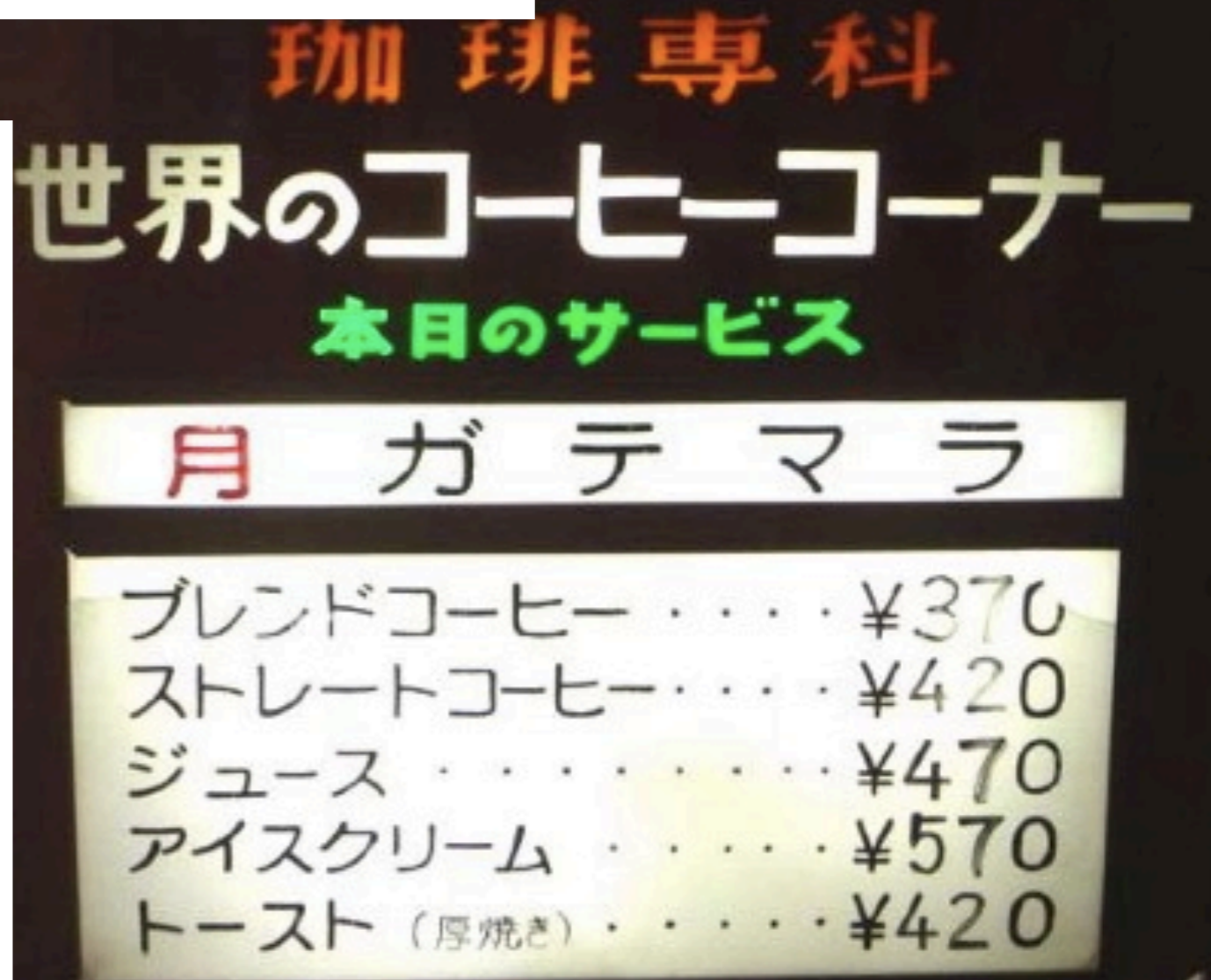
- コンピューター
- ko n py u - ta -
- kompyuuta (uu=û)
- computer
- アイスクリーム
- a i su ku ri - mu
- aisukuriimu
- ice cream
- アンドリュー・ビタビ
- ヨーグルト
- andoryuubitabi
- yo - gu ru to
- Andrew Viterbi
- yogurt

Katakana on Streets of Tokyo

Japanese just transliterates almost everything
(even though its syllable inventory is really small...)
but... it is quite easy for English speakers to decode
.... if you have a good language model!

from Knight & Sproat 09

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



More Japanese Transliterations

- rapputoppu ラップトップ ● laptop
- bideoteepu ビデオテープ ● video tape
- shoppingusentaa ショッピングセンター ● shopping center
- shiitoberuto シートベルト ● seat belt
- chairudoshiito チャイルトシート ● child seat
- andoryuubitabi アンドリュー・ビタビ ● Andrew Viterbi
- bitabiarugorizumu ビタビアルゴリズム ● Viterbi Algorithm

(hw3) Katakana => English

- your job in HW3: decode Japanese Katakana words (transcribed in Romaji) back to English words
- koohiikoonaa => coffee corner



(hw3) Katakana => English

- Decoding
 - really decipherment!
- what about duplicate strings?
 - from different paths in WFST!
 - n-best cruching, or...
 - weighted determinisation
 - see extra reading on course website for Mohri+Riley paper



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 => アイス クリーム => ice cream
 - how about “gaaruhurendo” and “shingururuumu” ?
- this is an even more important issue in Chinese
 - 南京市长江大桥
- also in other East Asian Languages
- also in English: sounds => words (speech recognition)

What if English were written as Chinese...

- thisisacoursetaughtinthefallsemesterofthisyearatusc
- actually, Latin used to be written exactly like this!
 - “scripta continua” => “interpuncts” (center dots) =>
- this might be a final project topic (on the easier side)

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

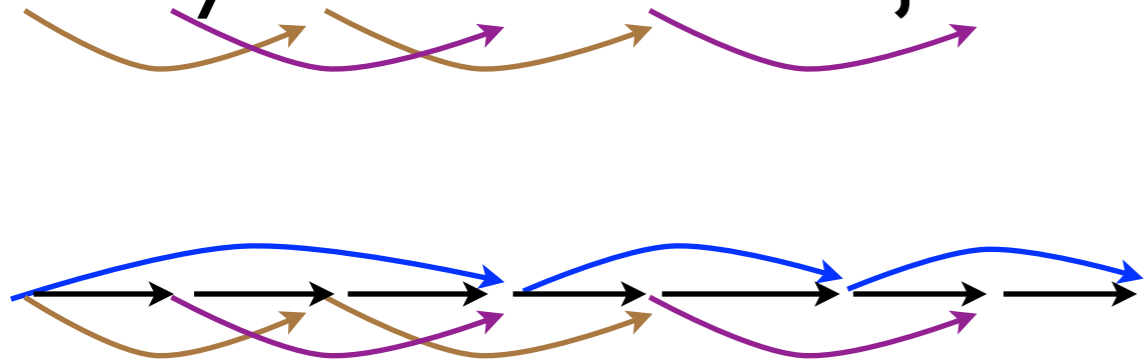
... - ... - people dominate-podium

“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

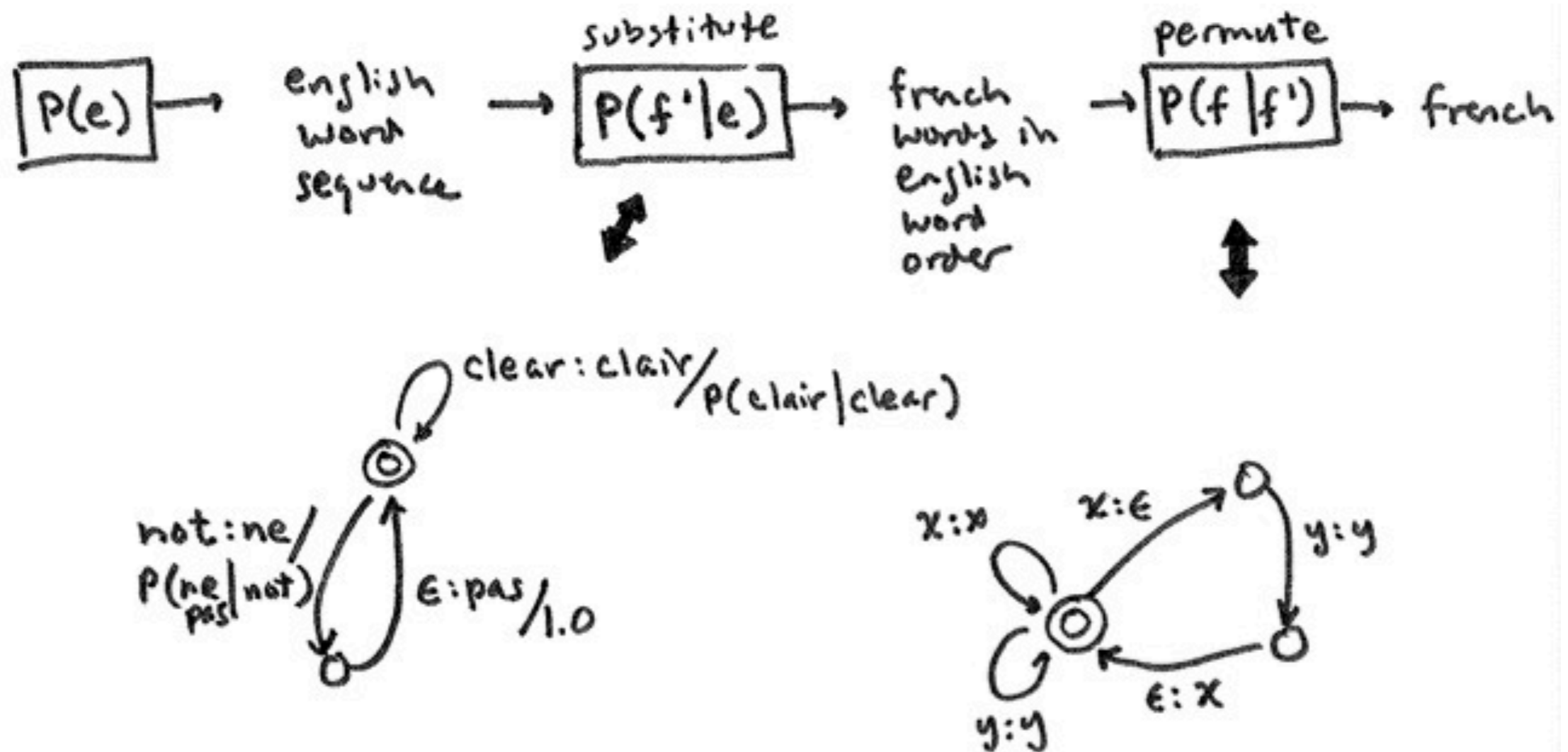
tagging problem

Word Segmentation Cascades

- a good idea for final project (Chinese/Japanese)

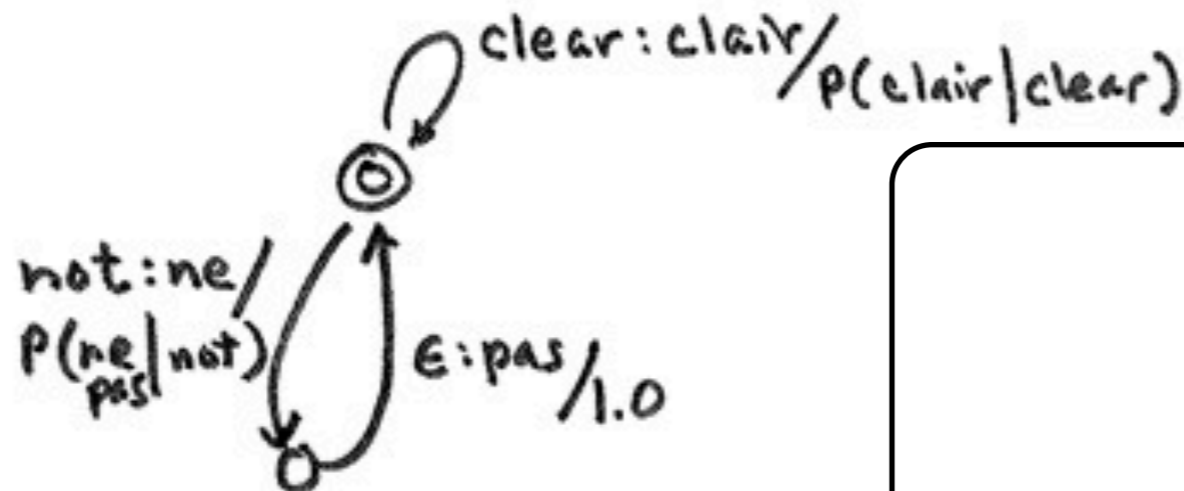
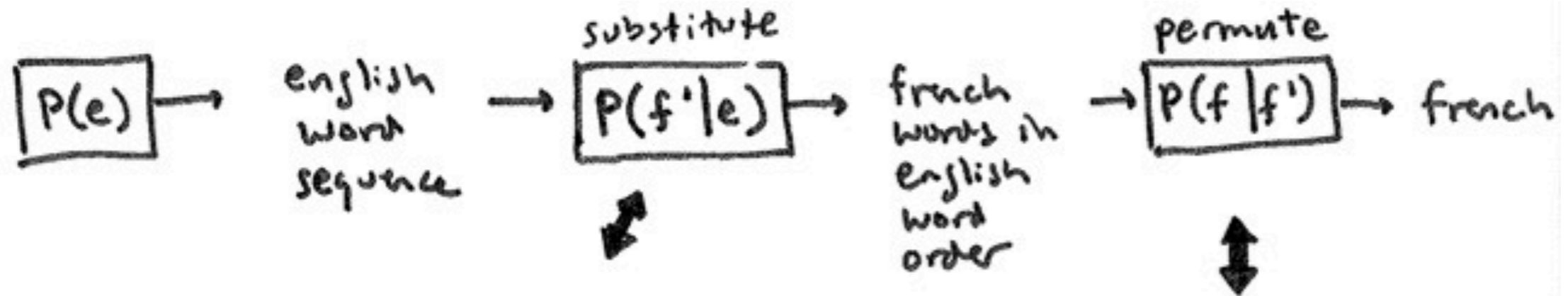
Machine Translation

- simplest model: word-substitution and permutation
- does it really work??



Machine Translation Permutation

- how would you model permutation in FSTs?

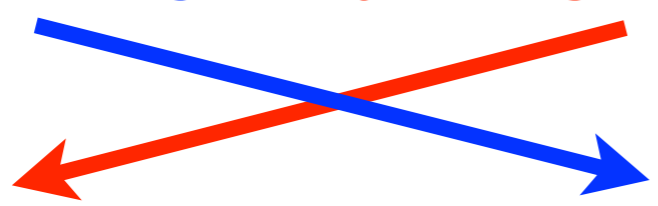


Phrase-based Decoding

与 沙龙 举行 了 会谈

yu Shalong juxing le huitan

held a talk with Sharon



with Sharon held talks

with Sharon held a talk

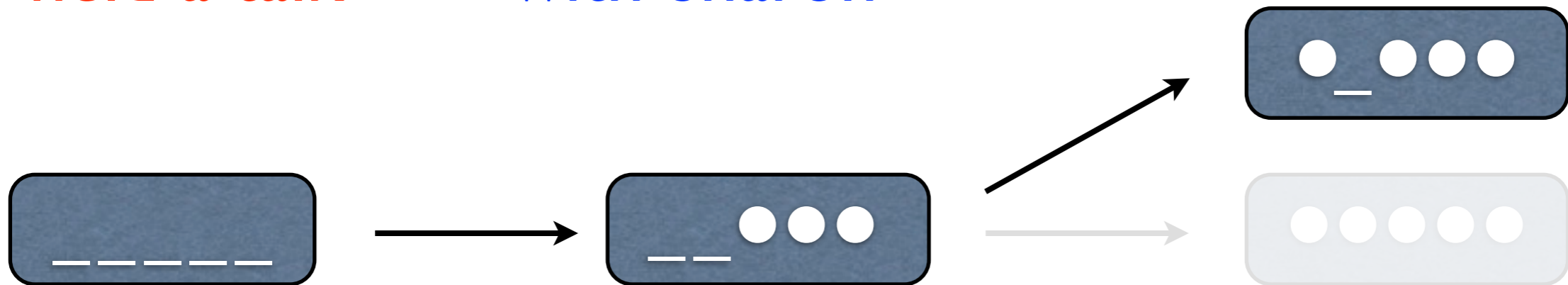
yu Shalong juxing le huitan

Phrase-based Decoding

与 沙龙 举行 了 会谈

yu Shalong juxing le huitan

held a talk with Sharon



with Sharon held talks

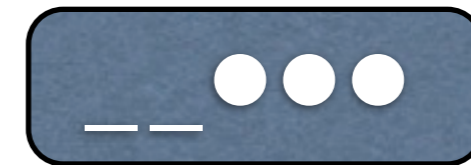
with Sharon held a talk

yu Shalong juxing le huitan

Phrase-based Decoding

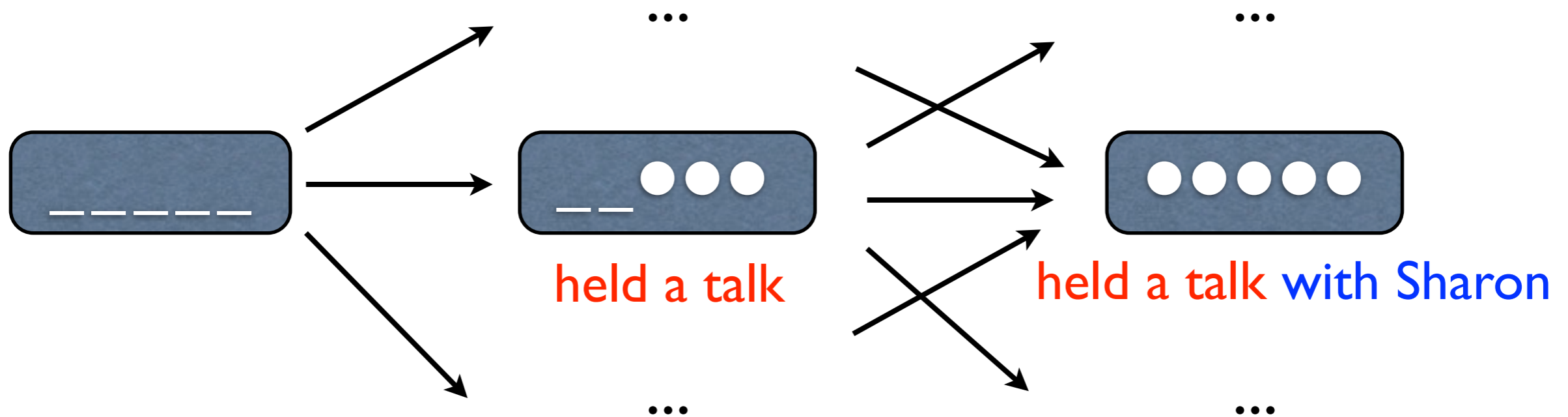
与 沙龙 举行 了 会谈
yu Shalong juxing le huitan
held a talk with Sharon

source-side: coverage vector



held a talk

target-side: grow hypotheses
strictly left-to-right



space: $O(2^n)$, time: $O(2^n n^2)$ -- cf. traveling salesman problem

Phrase-based Cascades

- english LM \Rightarrow (english) \Rightarrow phrase substitutions (n^2)
 \Rightarrow (foreign phrases in english word order)
 \Rightarrow permutations (2^n) \Rightarrow (foreign)
- a good idea for final project (on the harder end)
- wait, where does the phrase table come from?
 - \Rightarrow word-aligned english-foreign sentence pairs

Traveling Salesman Problem & MT

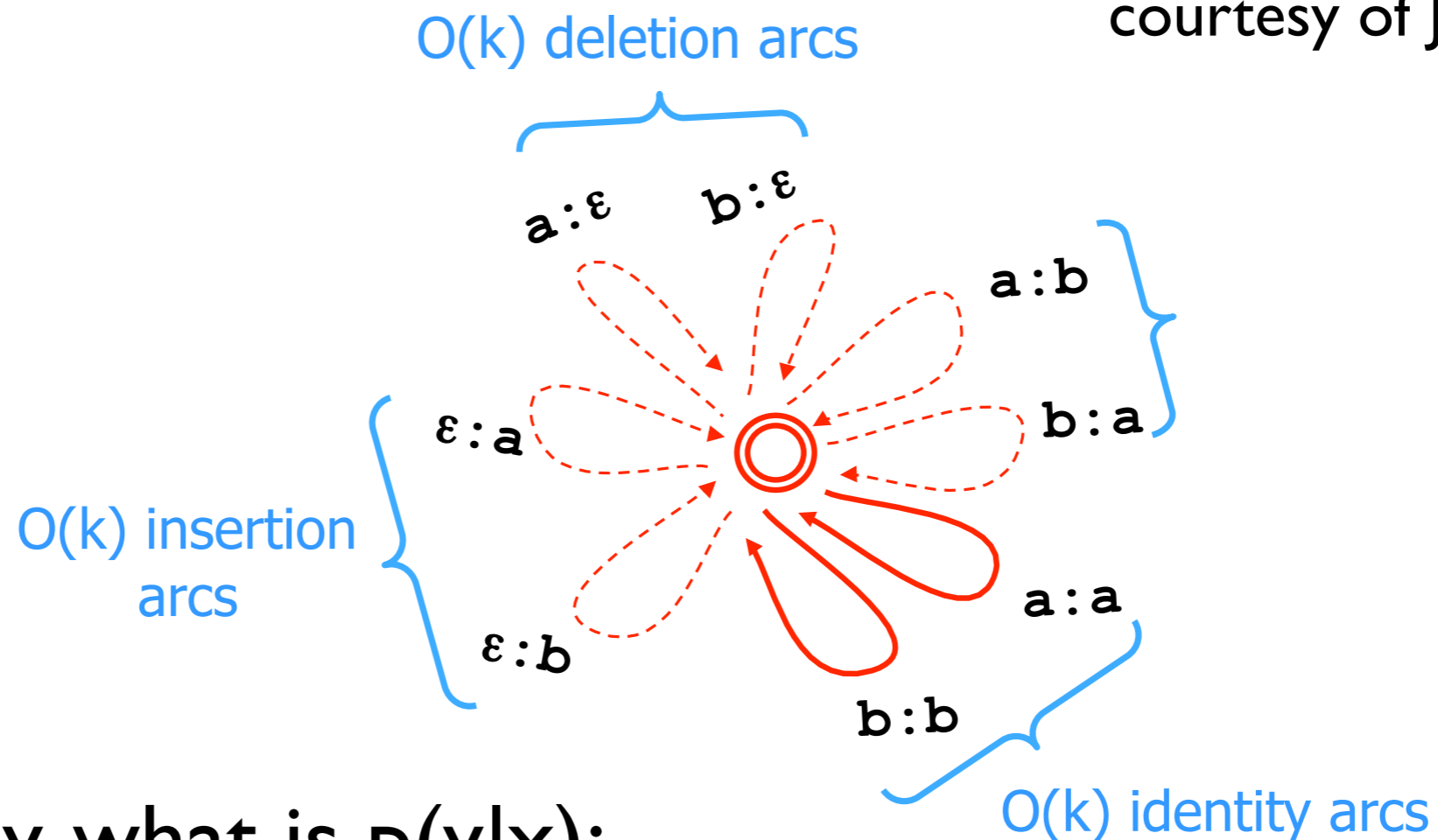
- a classical NP-hard problem
 - goal: visit each city once and only once
- exponential-time dynamic programming
 - state: cities visited so far (bit-vector)
 - search in this $O(2^n)$ transformed graph
- MT: each city is a source-language word
 - restrictions in reordering can reduce complexity => distortion limit
 - => syntax-based MT



(Held and Karp, 1962; Knight, 1999)

Example: Edit Distance

courtesy of Jason Eisner



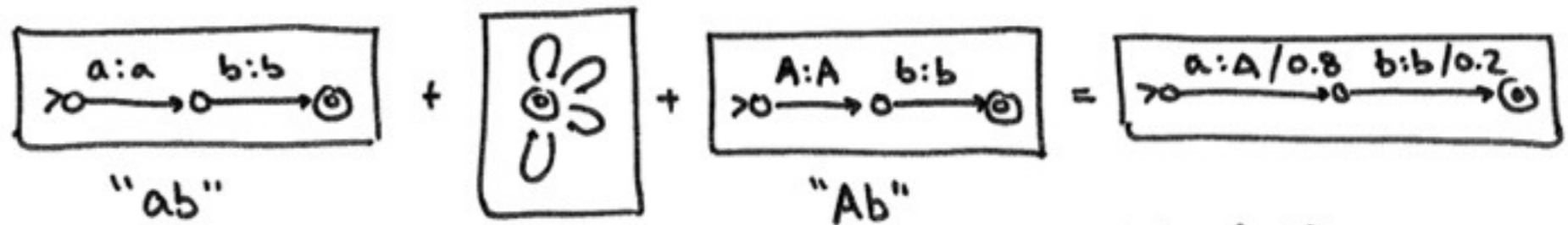
- a) given x, y , what is $p(y|x)$;
- b) what is the most likely seq. of operations?
- c) given x , what is the most likely output y ?
- d) given y , what is the most likely input x (with LM) ?

Edit Distance can model...

- part-of-speech tagging
- transliteration
- sound-spelling conversion
- word-segmentation

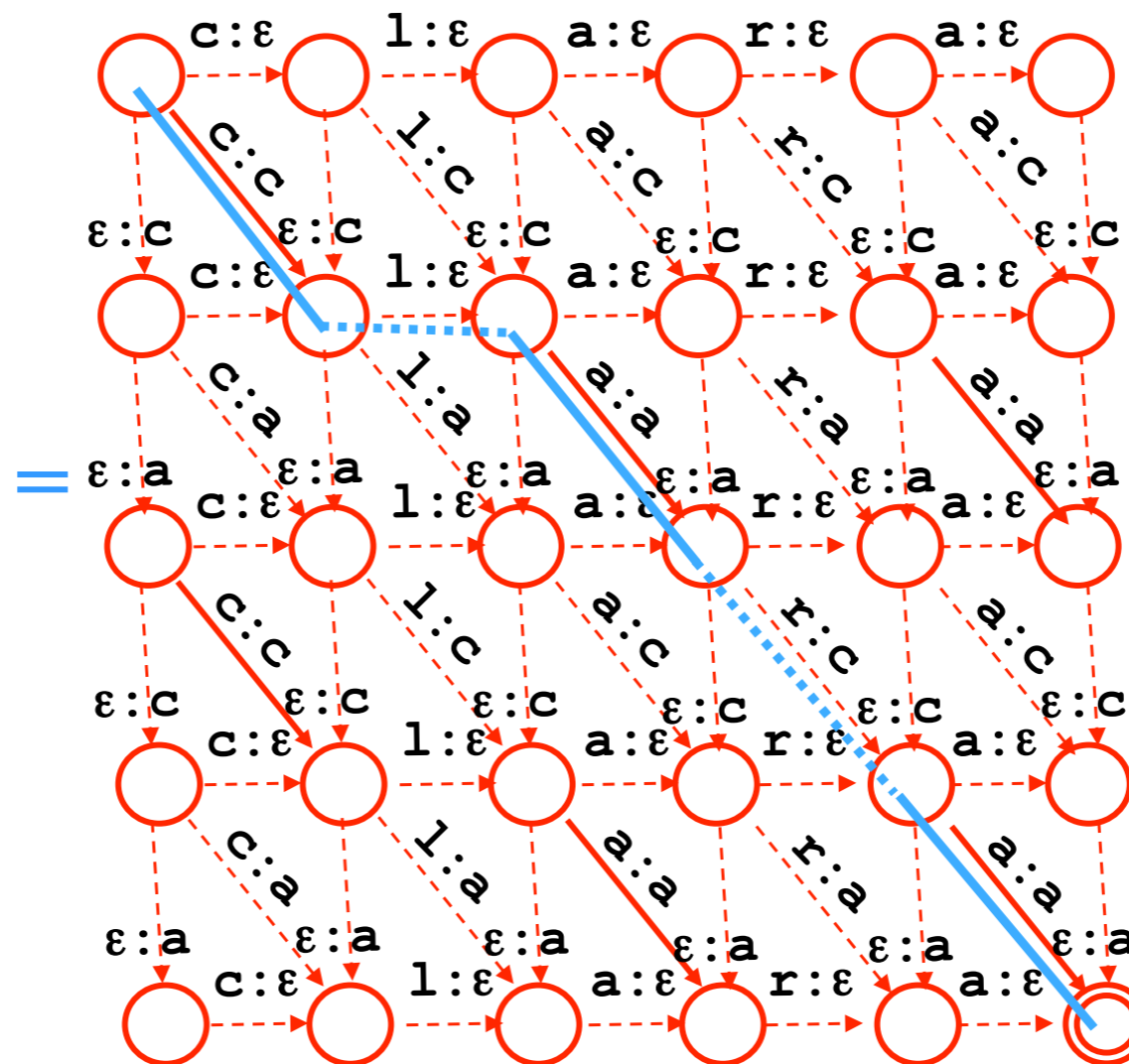
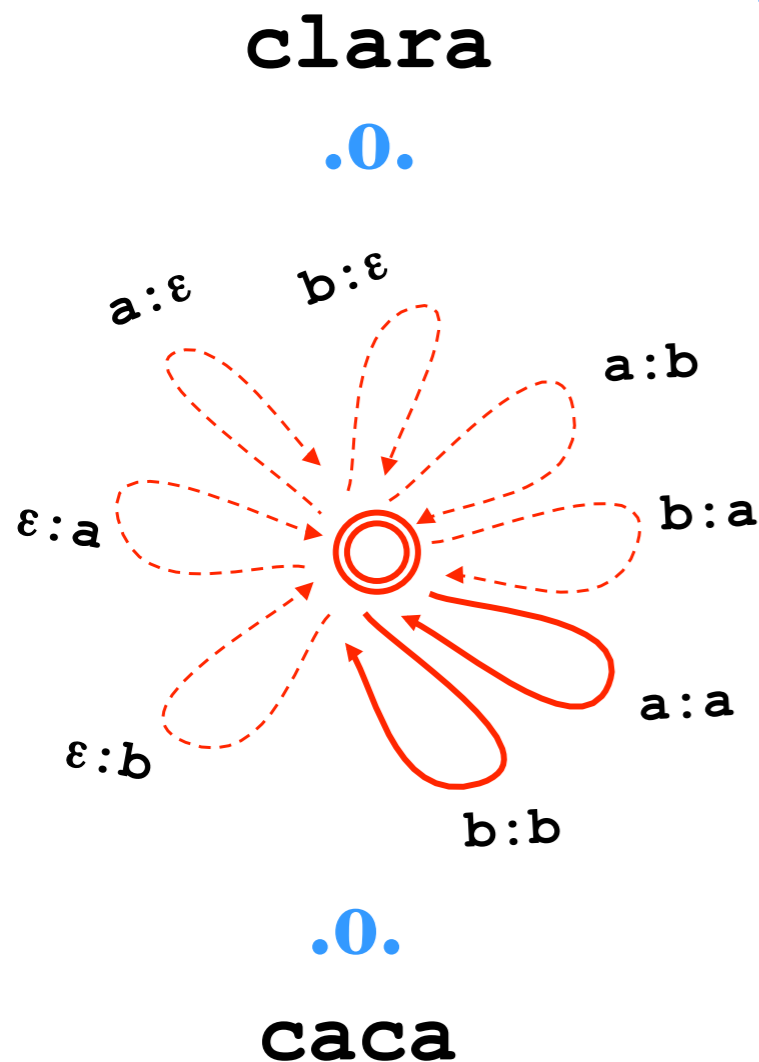
Given x and y...

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

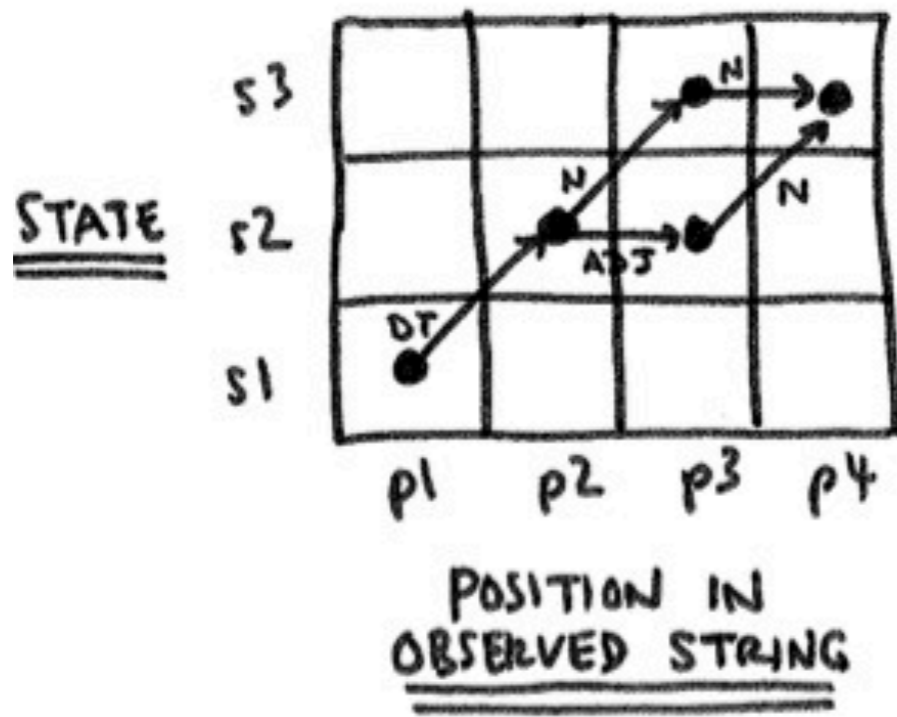
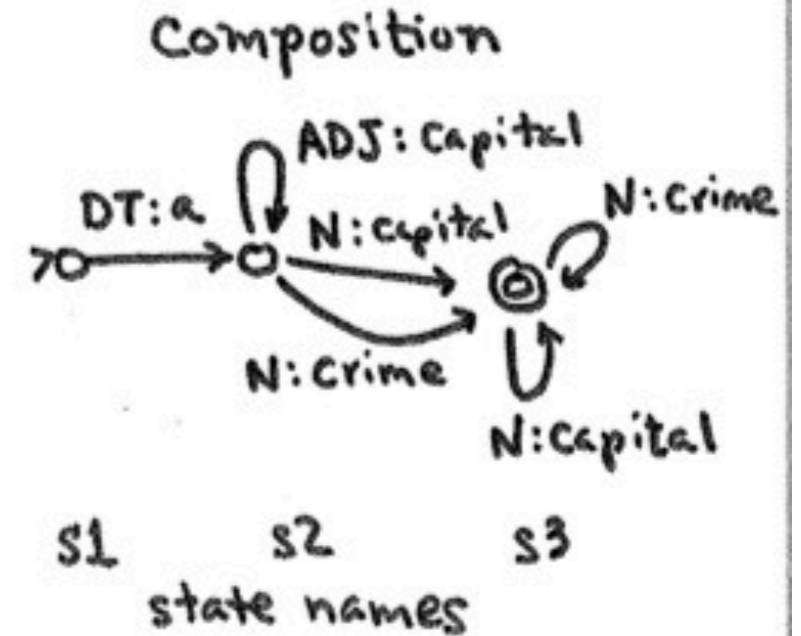
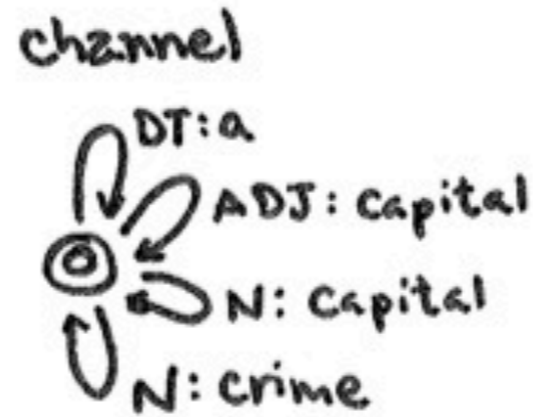
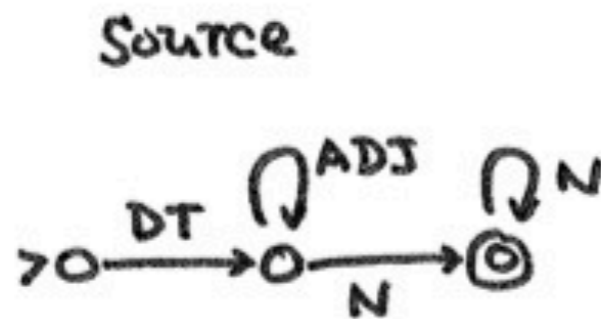


Best path (by Dijkstra's algorithm)

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



Example: General Tagging



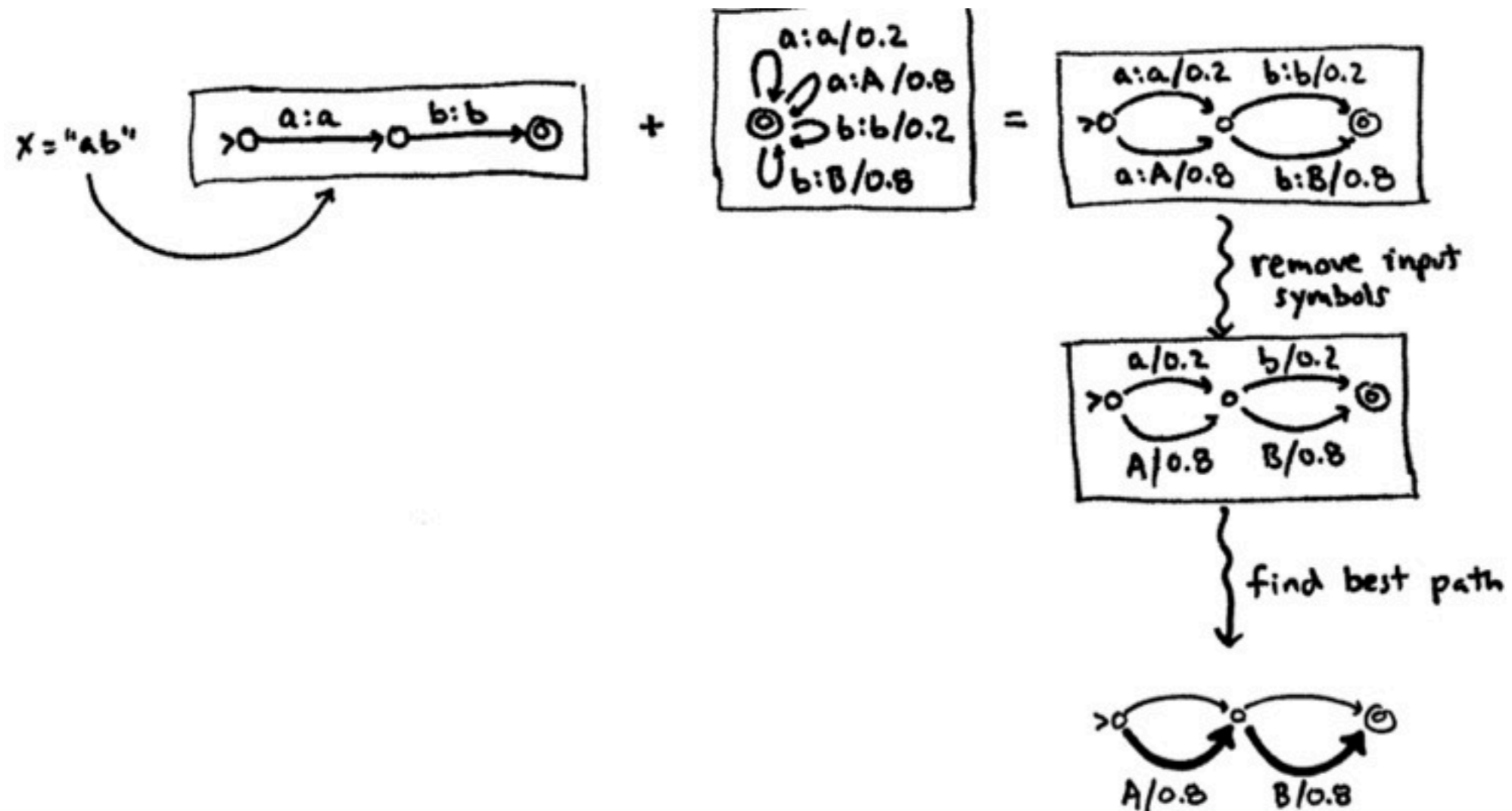
" a capital crime "

p1 p2 p3 p4

store $Q[i,j]$ best score to here
 $\psi[i,j]$ backpointer to best pred
 $\alpha[i,j]$ sum of scores to here

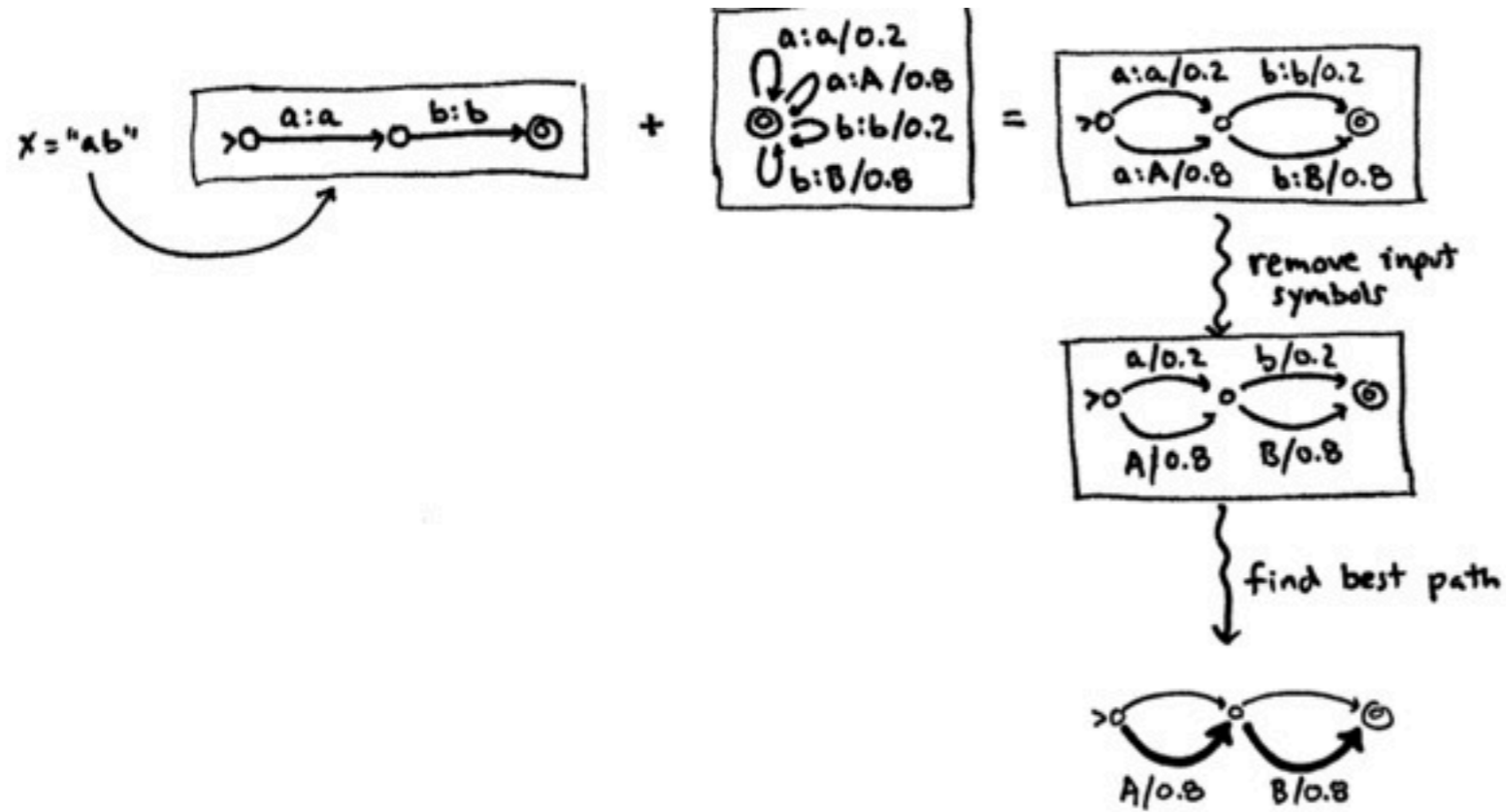
Most Likely “Corrupted Output”

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



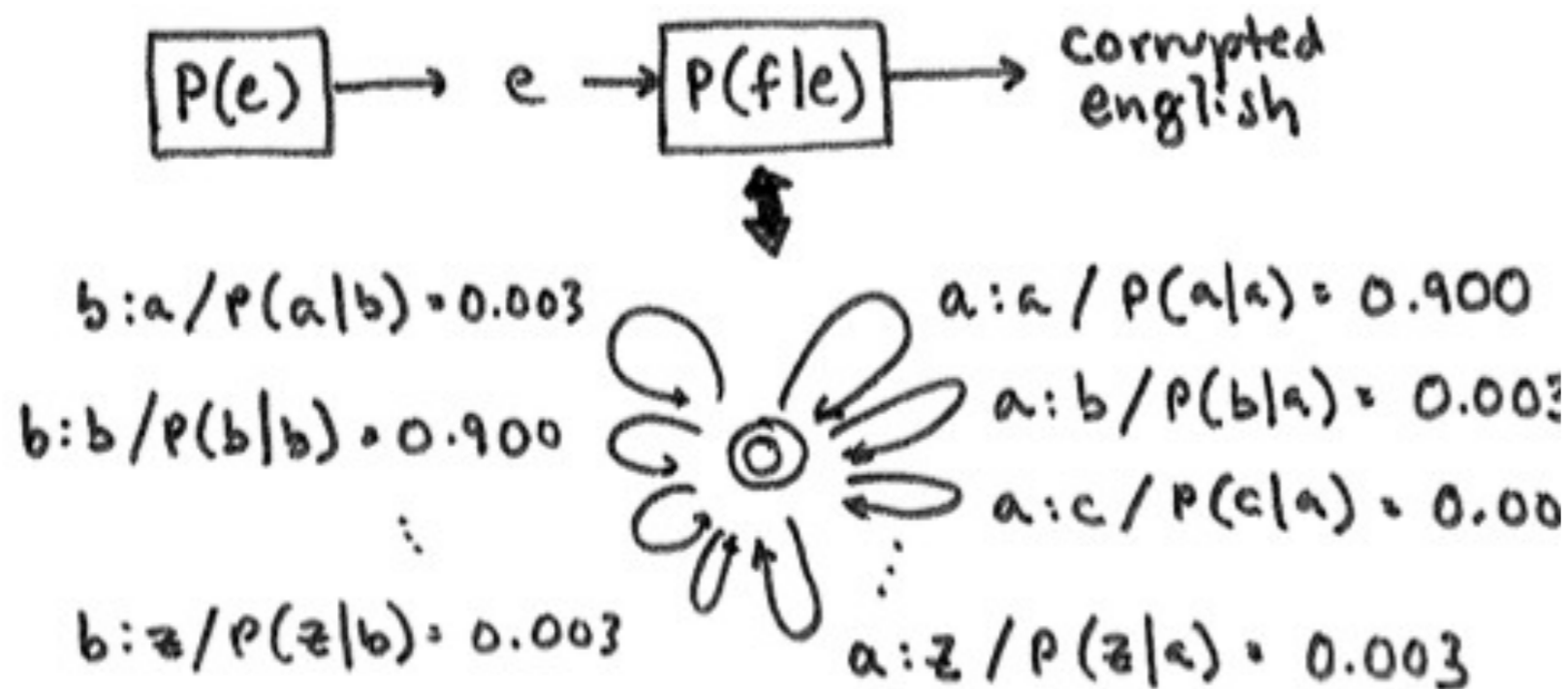
$$\text{so, } \underset{y}{\operatorname{argmax}} P(y | ab) = AB$$

DP for “most likely corrupted”



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

d) Most Likely “Original Input”

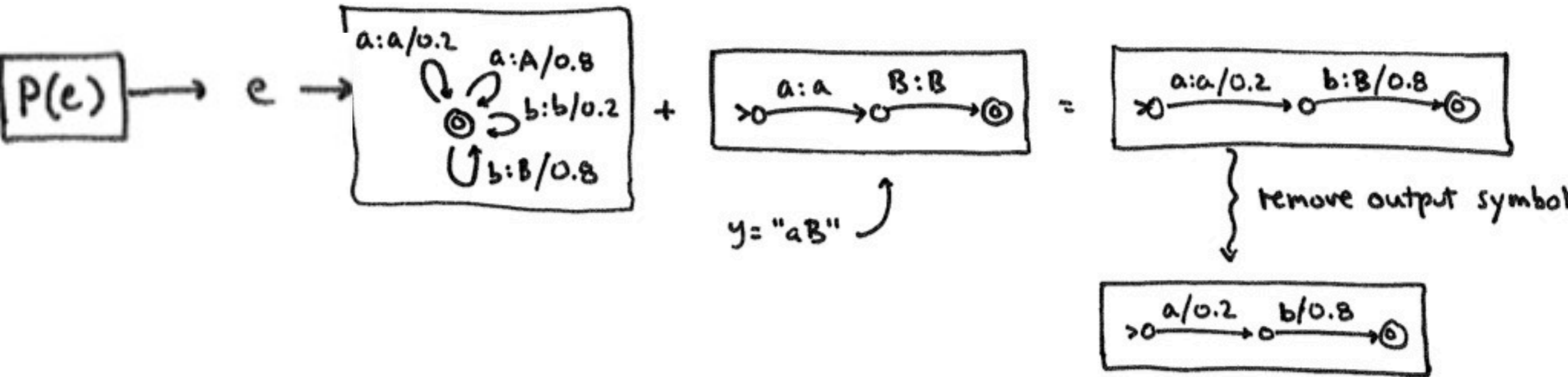


- ignores insertions/deletions
- similar to "bad OCR" channel

- 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?

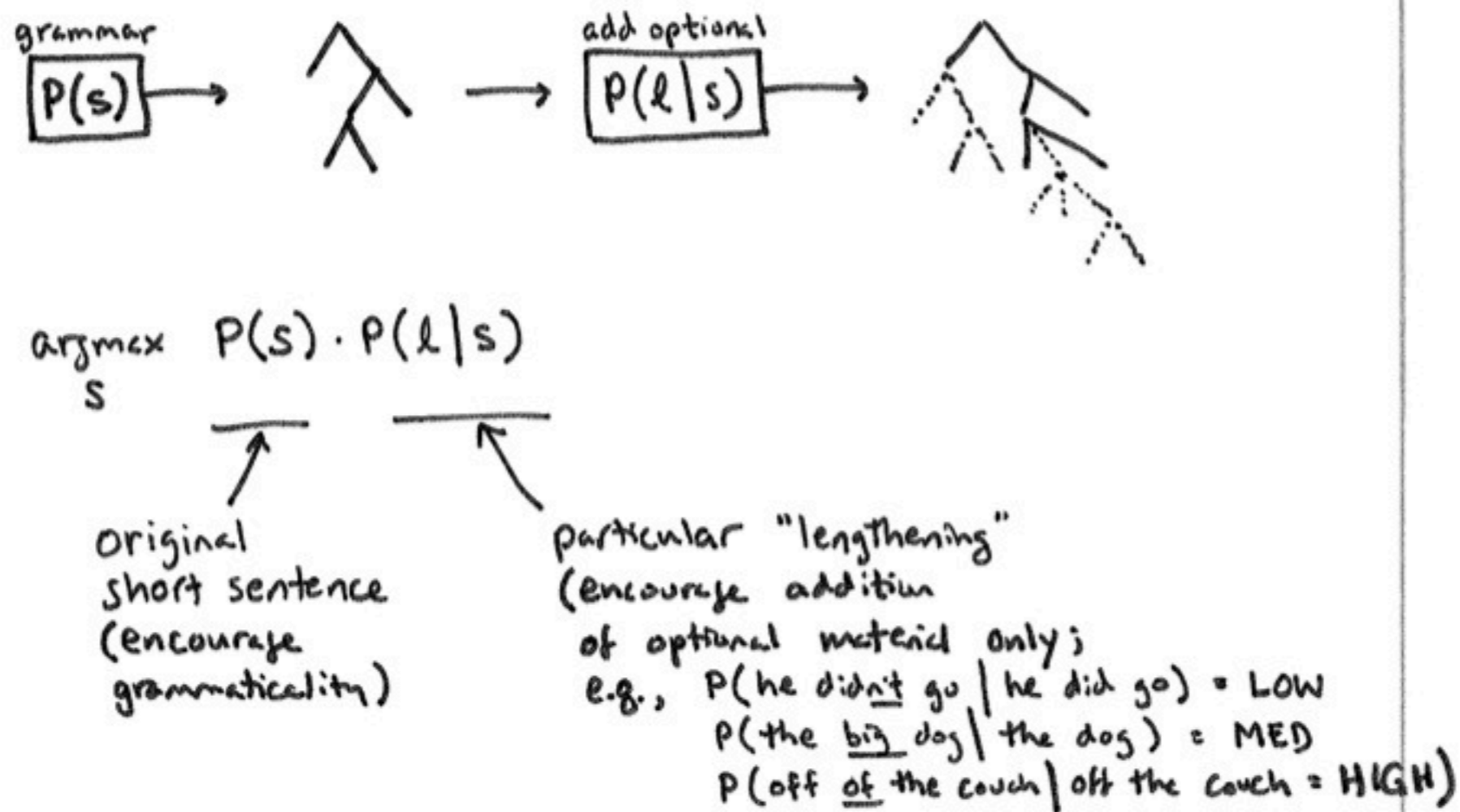
Dynamic Programming for d)

- given y , what is the most likely x with $\max p(x) p(y|x)$



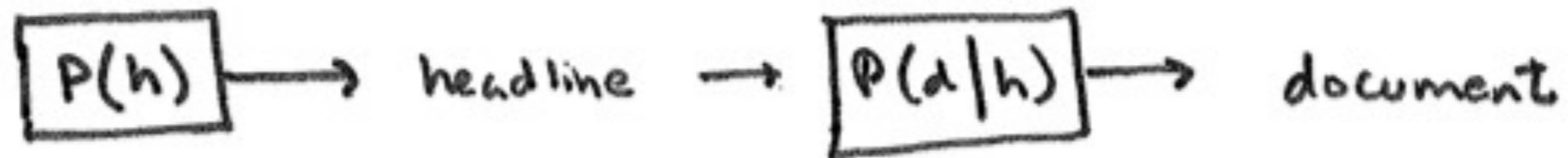
Beyond Finite-State Models

- sentence summarization



Beyond Finite-State Models

- headline generation



$$\operatorname{argmax}_h P(h) \cdot P(d|h)$$

looks like a proper headline

if this were a headline, d would be a reasonable document to go with it (i.e., d fleshes out h).

Beyond Finite-State Models

- information retrieval



used to rank documents, not construct new ones!

query may contain words not in document.

Beyond Finite-State Models

- parsing