CS-AD 220 – Spring 2016

Natural Language Processing

Session 15: 29-Mar-16

Prof. Nizar Habash

Some slides are adapted from Jurafsky and Martin's course slides on Speech and Language Processing

Moving Legislative Day Class

- Spring Break is March 18 – 25, 2016
- Sat March 26, 2016 is a Legislative *Thursday*
- Move to

Sat April 2, 2016 at 10am

Same Classroom C2-E049

POS Tagging

Assign the correct POS tag in context!

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

POS Tagging
Assign the correct POS tag in context!

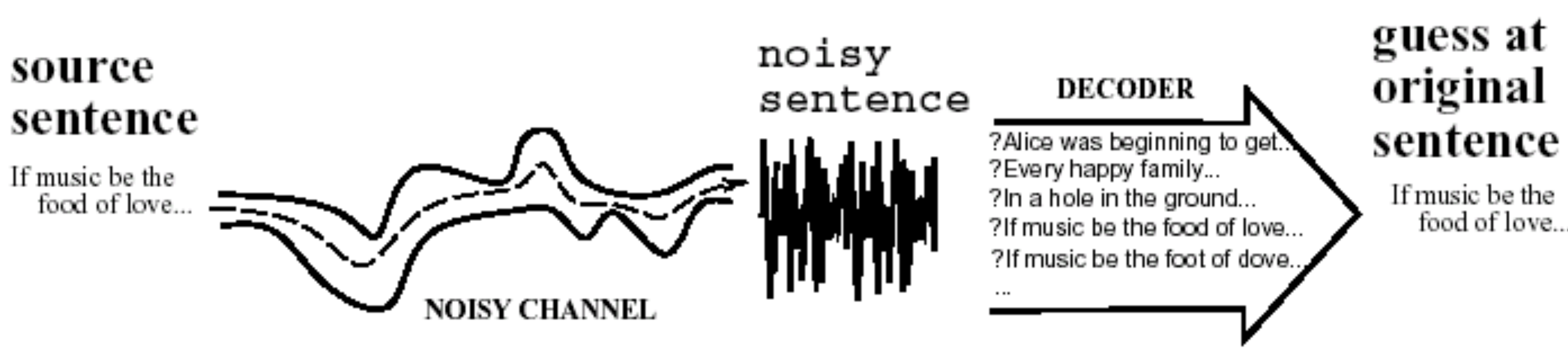
			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

Two Methods for POS Tagging

1. Rule-based tagging
   - Large databases of hand-written rules
     - EngCG / ENGTWOL
2. Stochastic
   - Probabilistic sequence models
     - HMM (Hidden Markov Model) tagging
     - …

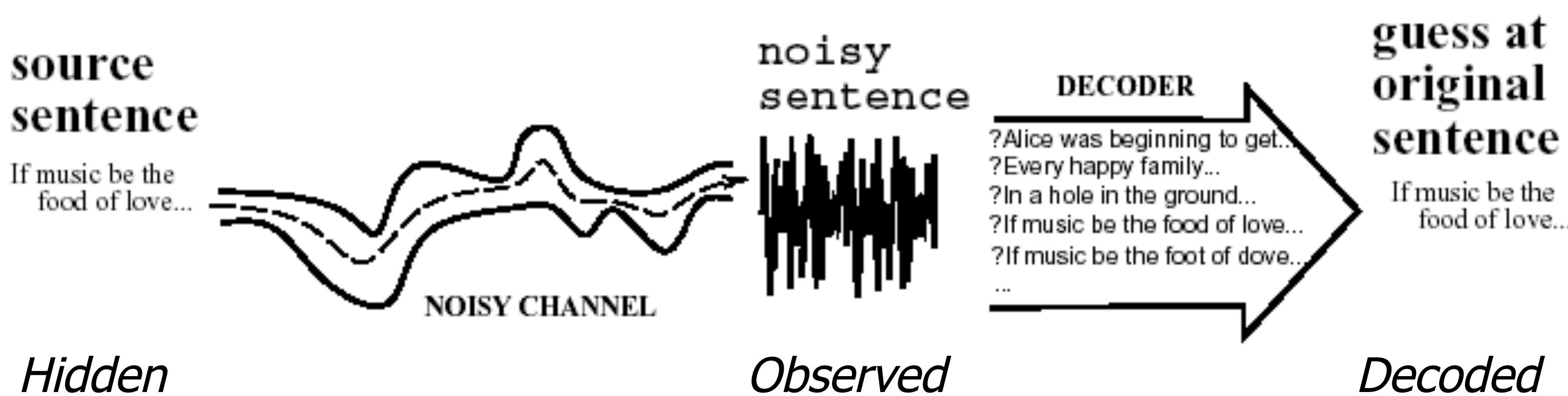
Hidden Markov Model Tagging

- Using an HMM to do POS tagging is a special case of *Bayesian inference*
  - Bayes Rule
  - Foundational work in computational linguistics
- It is also related to the "noisy channel" model that's the basis for ASR, OCR and MT



HMMs and their Usage

- Speech recognition (observed: acoustic signal, hidden: words)
- Handwriting recognition (observed: image, hidden: words)
- Part-of-speech tagging (observed: words, hidden: part-of-speech tags)
- Machine translation (observed: foreign words, hidden: words in target language)



From Natural Language Processing Slides by Bonnie Dorr and Christof Monz

POS Tagging as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags (the "hidden" sequence") that corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Getting to HMMs

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Hat ^ means "our estimate of the best one"
- Argmax_x f(x) means "the x such that f(x) is maximized"

Getting to HMMs

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n|w_1^n)$$

- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform this equation into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

Likelihood and Prior

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \; \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \;\; \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n|t_1^n) \; \approx \; \prod_{i=1}^{n} P(w_i|t_i)$$

$$P(t_1^n) \; \approx \; \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \, P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1})$$

Two Kinds of Probabilities

- Tag transition probabilities $p(t_i|t_{i-1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be low.
  - Compute P(NN|DT) by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two Kinds of Probabilities

- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P(is|VBZ) by counting in a labeled corpus:

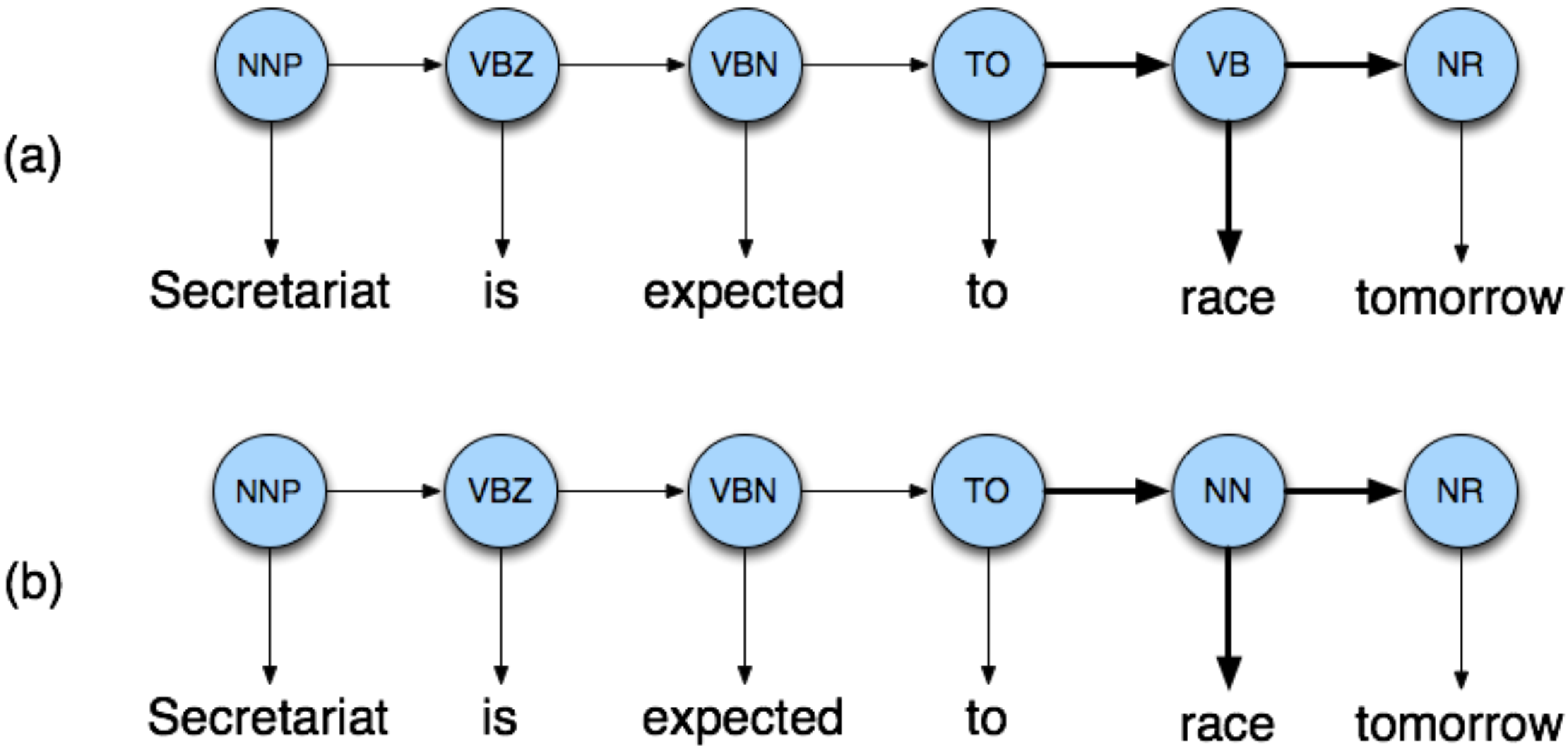
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Example: The Verb “race”

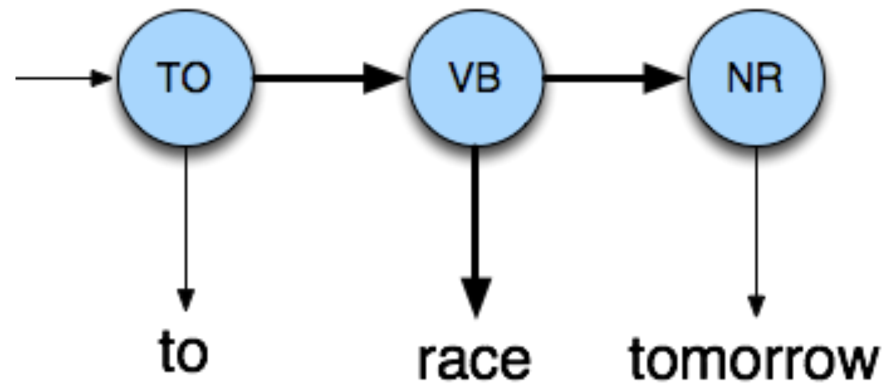
- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

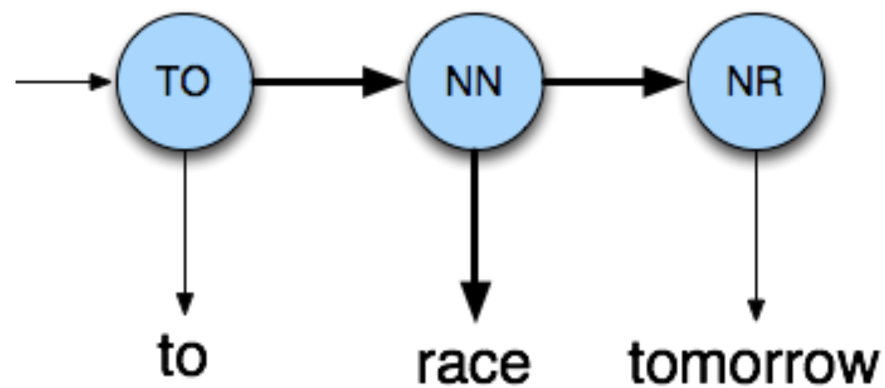
Disambiguating “race”



Disambiguating “race”

- P(VB|TO) = .83
- P(NN|TO) = .00047
- P(race|VB) = .00012
- P(race|NN) = .00057
- P(NR|VB) = .0027
- P(NR|NN) = .0012
- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb reading





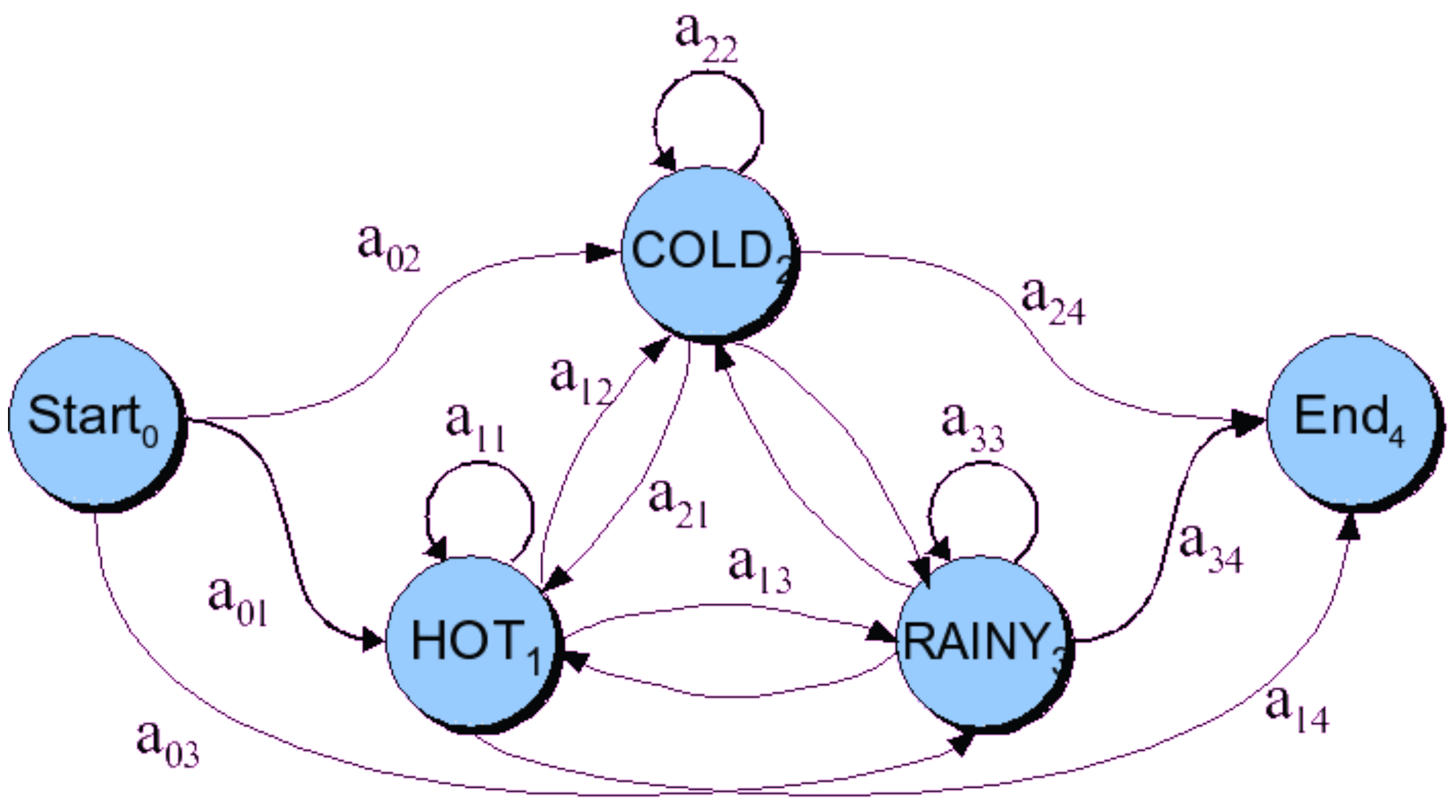
Hidden Markov Models

- What we’ve described with these two kinds of probabilities is a Hidden Markov Model (HMM)

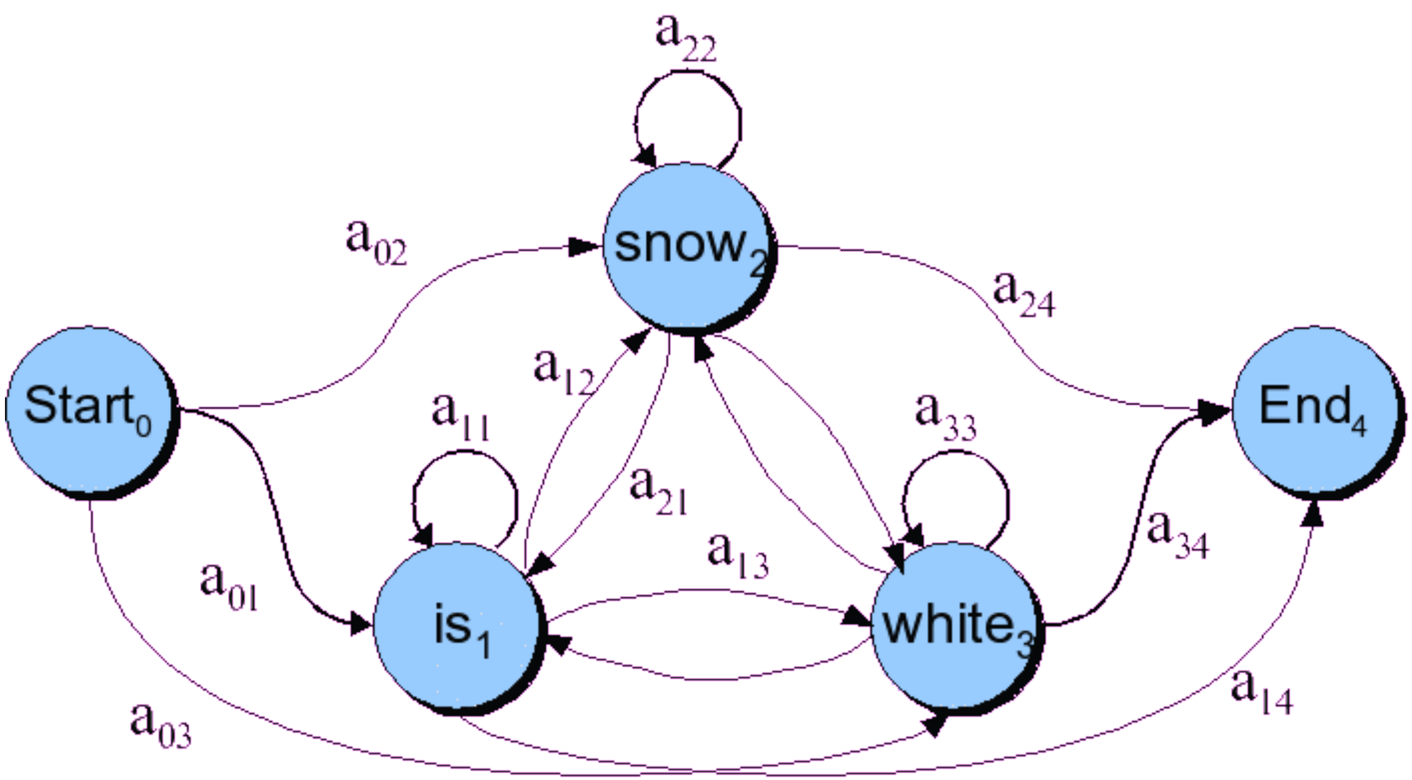
Definitions

- A weighted finite-state automaton (WFST) adds probabilities to the arcs
  - The sum of the probabilities on arcs leaving any node must sum to one
- A Markov chain is a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can’t represent inherently ambiguous problems
  - Assigning probabilities to unambiguous observed sequences
  - Conditioning on previously observed events
- Hidden Markov Models allows to talk about both observed events (words) and hidden events (POS tags)

Markov Chain for Weather



Markov Chain for Words



Markov Chain: "First-order observable Markov Model"

- A set of states
  - $Q = q_1, q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities $A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A

- Current state only depends on previous state

$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

Markov Chain for Weather

- What is the probability of 4 consecutive rainy days?
- Sequence is rainy-rainy-rainy-rainy
- I.e., state sequence is 3-3-3-3
- P(3,3,3,3) =
  - $\pi_1 a_{11} a_{11} a_{11} a_{11} = 0.2 \times (0.6)^3 = 0.0432$

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
  - See **hot** weather: we're in state **hot**
- But in part-of-speech tagging (and other things)
  - The output symbols are **words**
  - But the hidden states are **part-of-speech tags**
- So we need an extension!
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means we don't know which state we are in.

Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_{N;}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

- Observations $O = o_1, o_2 \ldots o_{T;}$
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Observation likelihoods / Emission probabilities
  - Output probability matrix $B = \{b_i(k)\}$

$$b_i(k) = P(X_t = o_k \mid q_t = i)$$

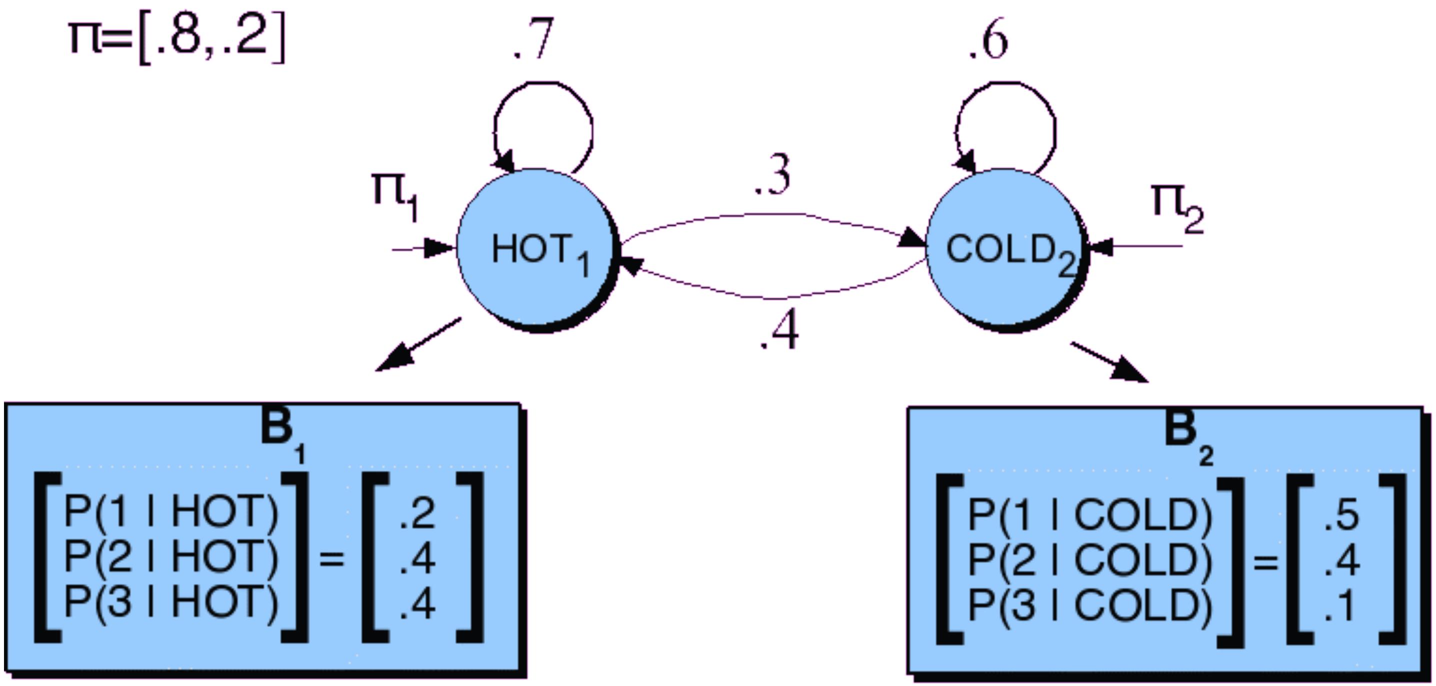
- Special initial probability vector π

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

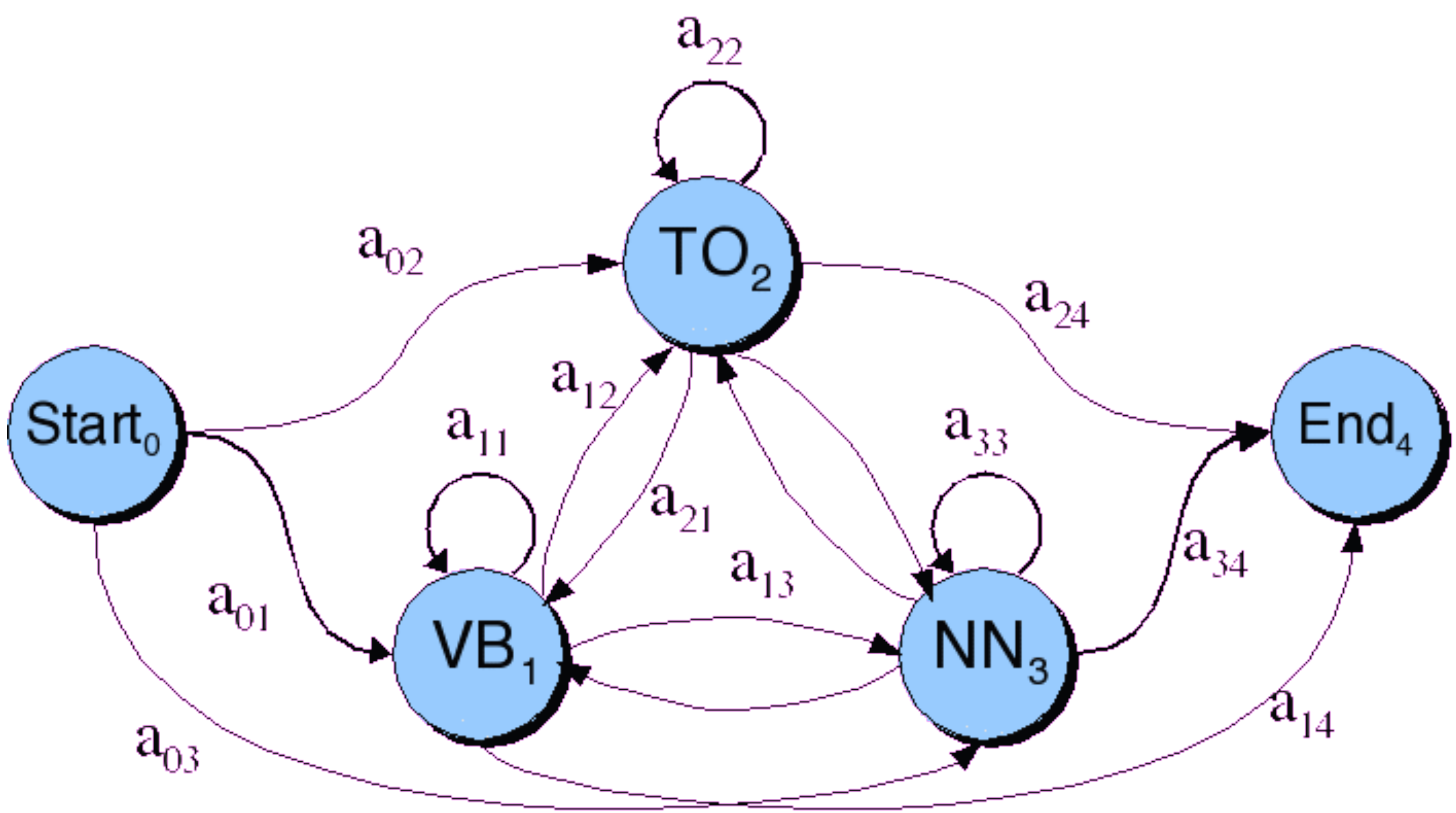
HMM for Weather

- Assume that we only have a record of the number of ice-creams eaten (say, 1, 2 or 3) (~ seen words) on each day; and we want to determine if the day was hot or cold (~ POS tags)
- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Weather Sequence: H,C,H,H,H,C…

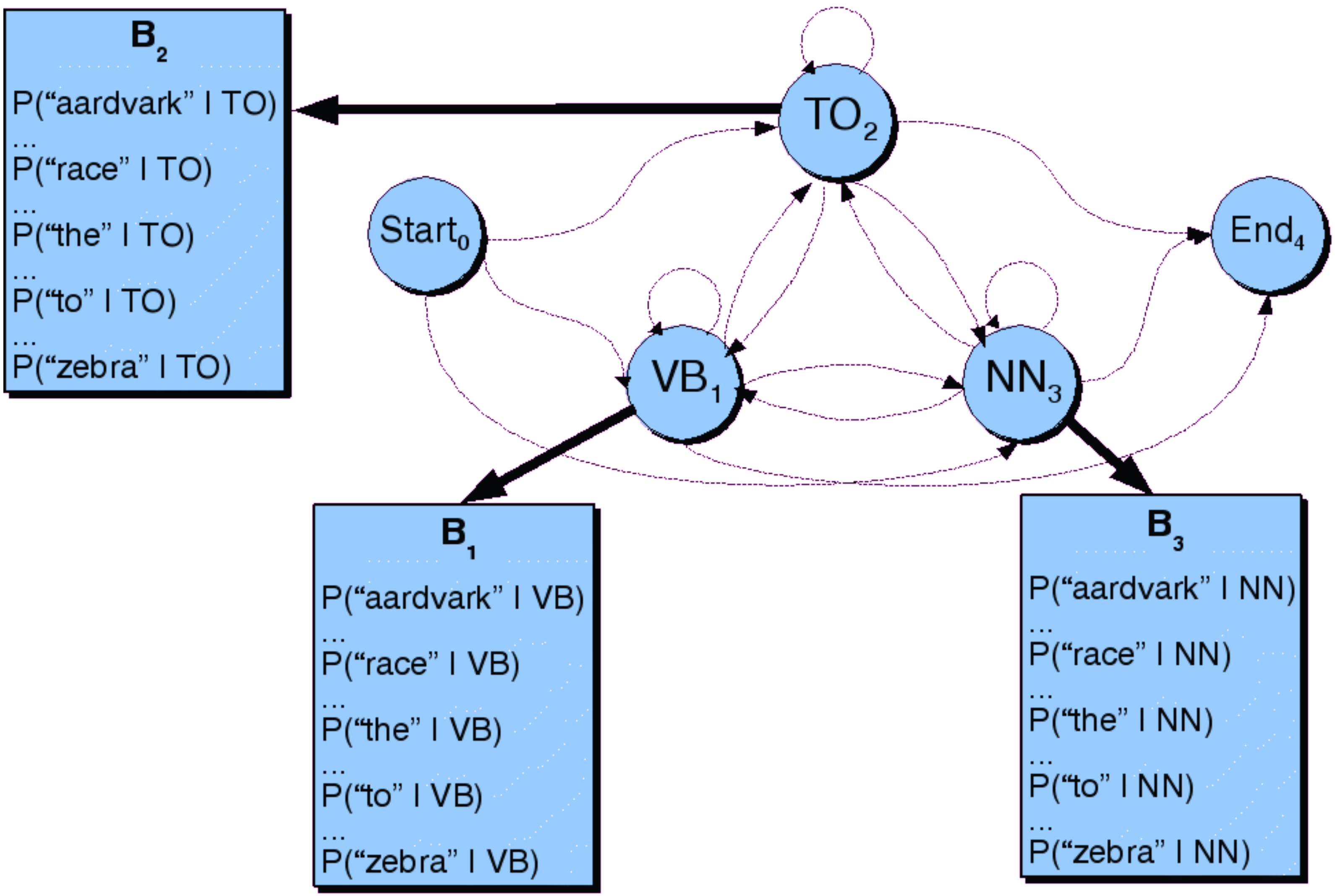
HMM for Ice Cream



Transition Probabilities



Observation Likelihoods



Decoding

- Ok, now we have a complete model that can give us what we need. Recall that we need to get

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

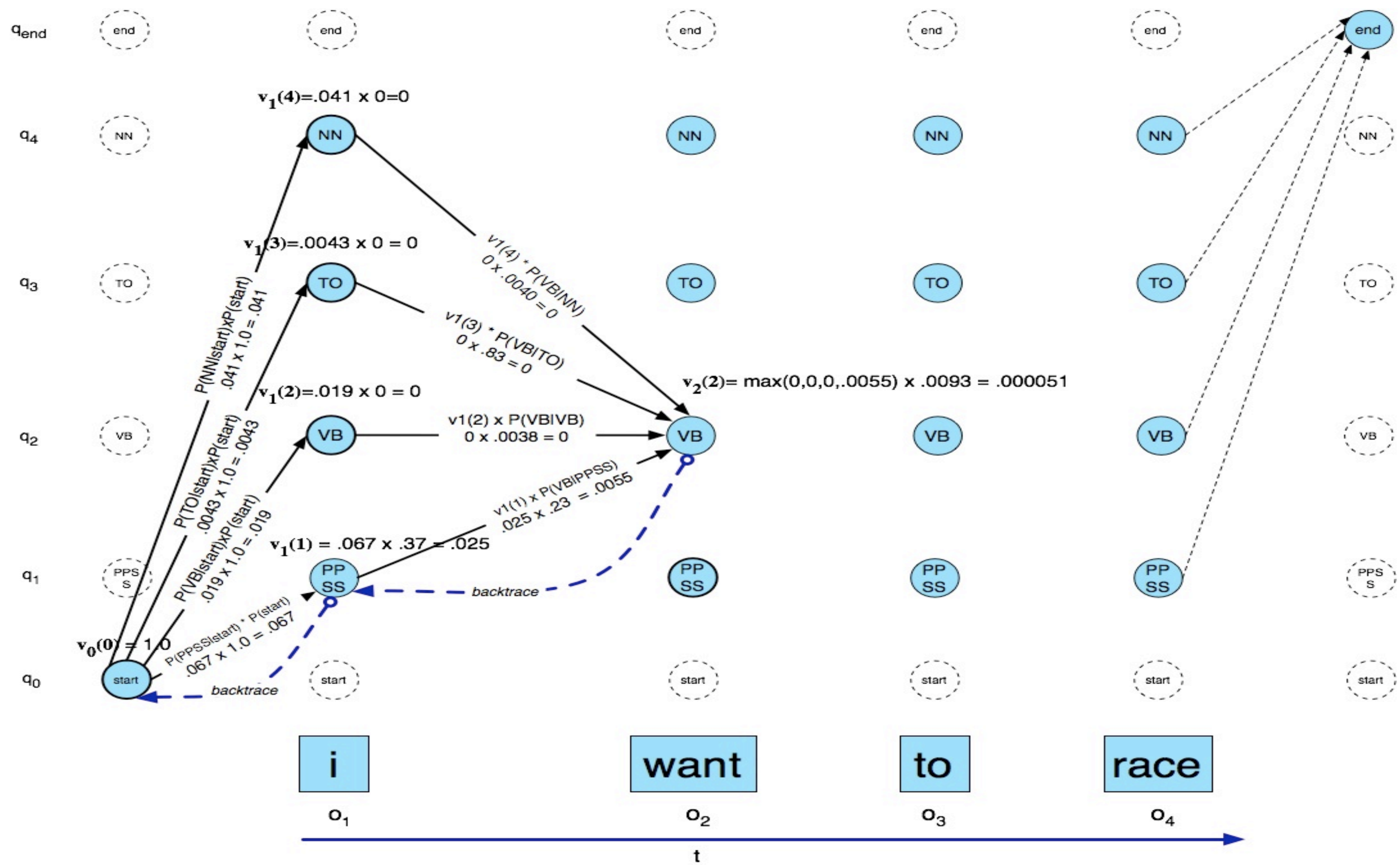
- We could just enumerate all paths given the input and use the model to assign probabilities to each.
  - Not a good idea.
  - Luckily dynamic programming (last seen in Ch. 3 with minimum edit distance) helps us here

The Viterbi Algorithm

function VITERBI(*observations* of len *T*,*state-graph*) **returns** *best-path*

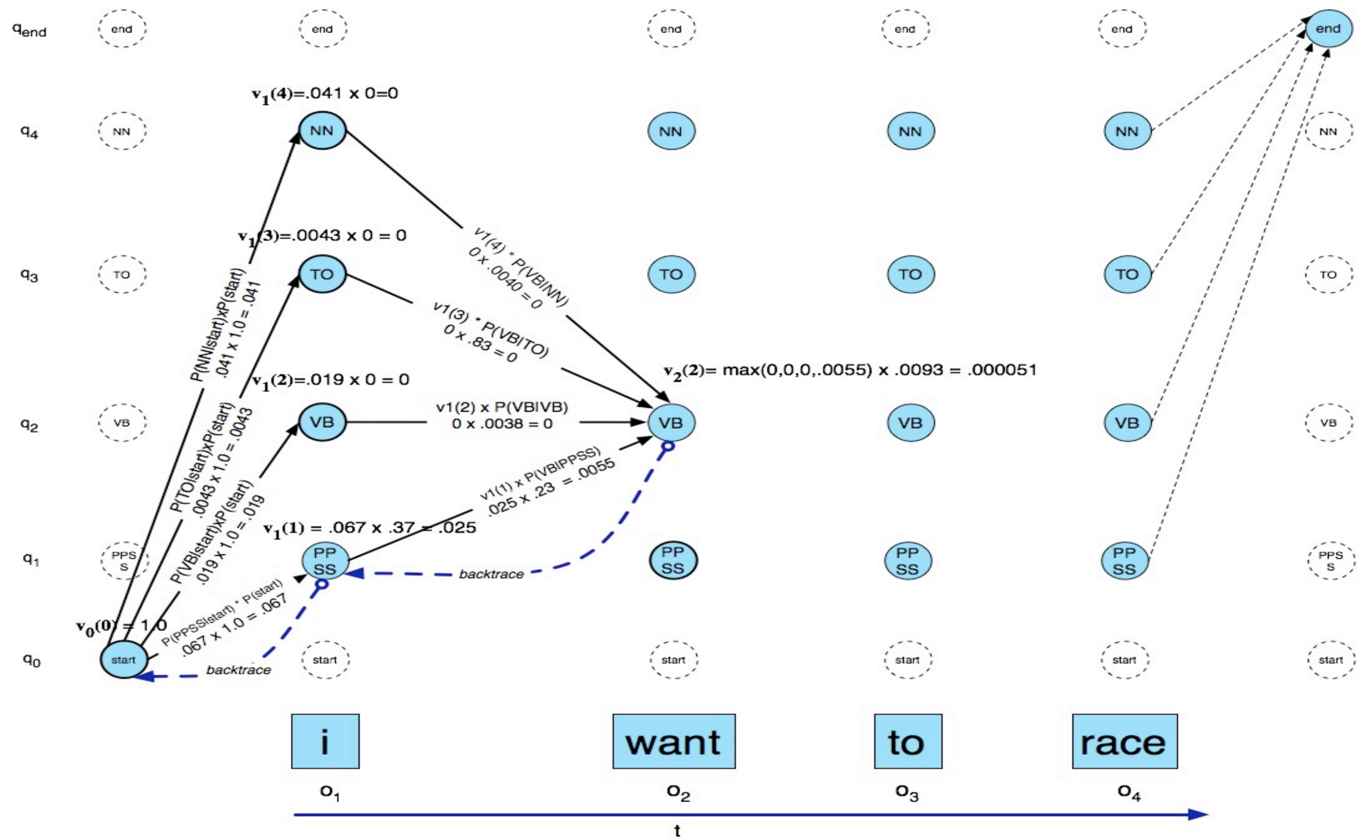
num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step *t* **from** 0 **to** *T* **do**
  for each state *s* **from** 0 **to** *num-states* **do**
    for each transition s' from *s* specified by *state-graph*
      new-score ← *viterbi*[*s*, *t*] * *a*[*s*,s'] * $b_{s'}(o_t)$
      if ((*viterbi*[s',*t+1*] = 0) || (*new-score* > *viterbi*[s', *t+1*]))
        then
          viterbi[s', *t+1*] ← *new-score*
          back-pointer[s', *t+1*] ← *s*
Backtrace from highest probability state in the final column of *viterbi[ ]* and return path

Viterbi Example

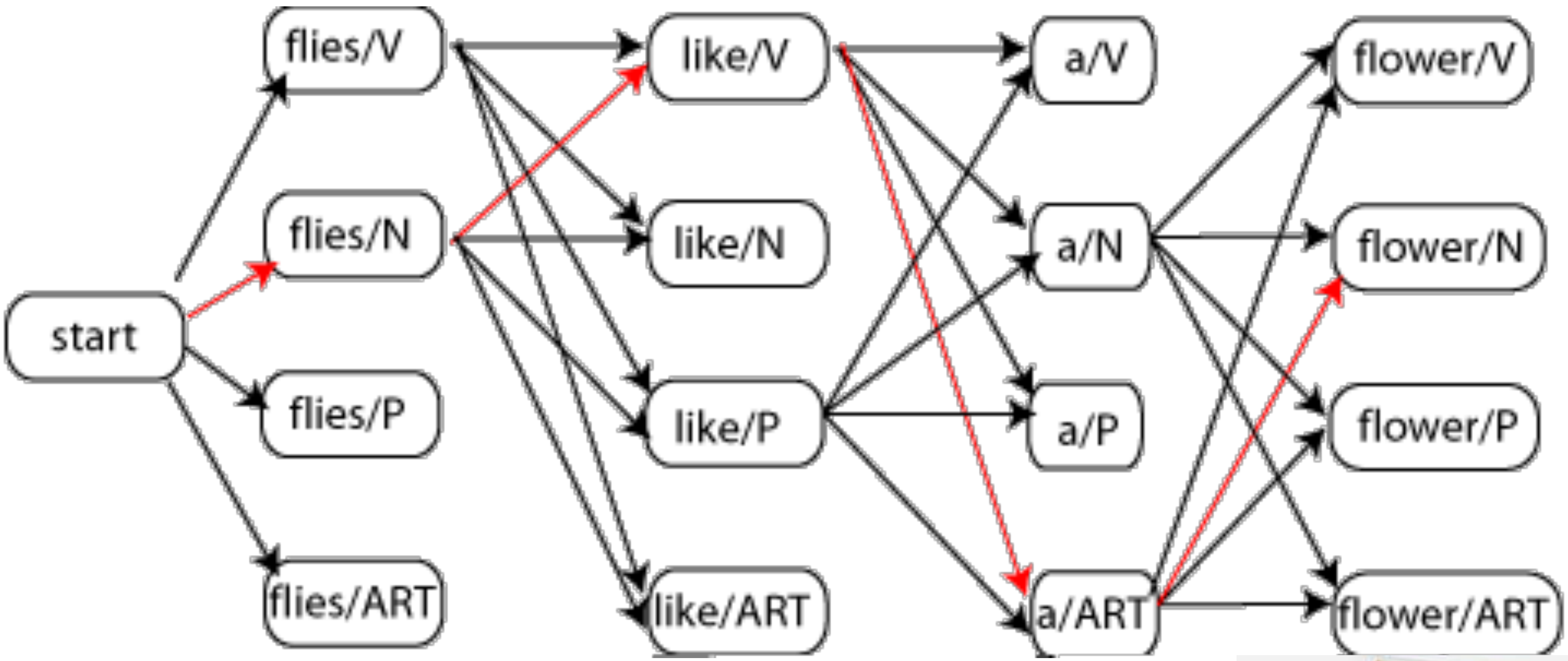


function VITERBI(*observations* of len *T*,*state-graph*) **returns** *best-path*

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step t **from** 0 **to** T **do**
  for each state s **from** 0 **to** *num-states* **do**
    for each transition s' from s specified by *state-graph*
      new-score ← *viterbi*$[s, t]$ * $a[s,s']$ * $b_{s'}(o_t)$
      if ((*viterbi*$[s',t+1] = 0$) || (*new-score* > *viterbi*$[s', t+1]$))
        then
          viterbi$[s', t+1]$ ← *new-score*
          back-pointer$[s', t+1]$ ← s
Backtrace from highest probability state in the final column of *viterbi[ ]* and return path



Viterbi Trellis



Garden trellis →

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob path to each cell, (not all paths).

Evaluation

- So once you have you POS tagger running how do you evaluate it?
  - Overall error rate with respect to a gold-standard test set.
  - Error rates on particular tags
  - Error rates on particular words
  - Tag confusions...

Error Analysis

- Look at a confusion matrix (gold, predicted)=(x,y)

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—

- *4.4% of all errors are caused by mistagging VBD as VBN*
- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)

Evaluation

- The result is compared with a manually coded "Gold Standard"
  - Typically accuracy reaches 96-97%
  - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.

Arabic POS Tagging
Morphological Ambiguity

- Morphological richness
  - Token Arabic/English = 80%
  - Type Arabic/English = 200%
- Morphological ambiguity
  - Each word: 12.3 analyses and 2.7 lemmas
- Derivational ambiguity
  - qAEdap: basis/principle/rule, military base, Qa'ida/Qaeda/Qaida

Morphological Ambiguity

- Inflectional ambiguity
  - *taktub*: you write, she writes
  - Segmentation ambiguity
    - wjd: *wajada* he found; *wa+jad~u*: and+grandfather

- Spelling ambiguity
  - Optional diacritics
    - kAtb: *kAtib* writer; *kAtab* to correspond
  - Suboptimal spelling
    - Hamza dropping: إ, أ → ا
    - Undotted ta-marbuta: ة → ه
    - Undotted final ya: ي → ى

Analysis vs. Disambiguation

بين

PV+PVSUFF_SUBJ:3MS	bay~an+a	He demonstrated
PV+PVSUFF_SUBJ:3FP	bay~an+~a	They demonstrated (f.p)
NOUN_PROP	biyn	Ben
ADJ	bay~in	Clear
PREP	bayn	Between, among

Morphological Analysis **is out-of-context**
Morphological Disambiguation is in-context

Analysis vs. **Disambiguation**

Will Ben Affleck be a good Batman?

هل سينجح بين أفليك في دور باتمان؟

	PV+PVSUFF_SUBJ:3MS	bay~an+a	He demonstrated
	PV+PVSUFF_SUBJ:3FP	bay~an+~a	They demonstrated (f.p)
*	**NOUN_PROP**	**biyn**	**Ben**
	ADJ	bay~in	Clear
	PREP	bayn	Between, among

Morphological Analysis is out-of-context

Morphological Disambiguation is in-context

Morphological Disambiguation *in English*

- Select a morphological tag that fully describes the morphology of a word
- Complete English morphological tag set (Penn Treebank): 48 tags

Verb:

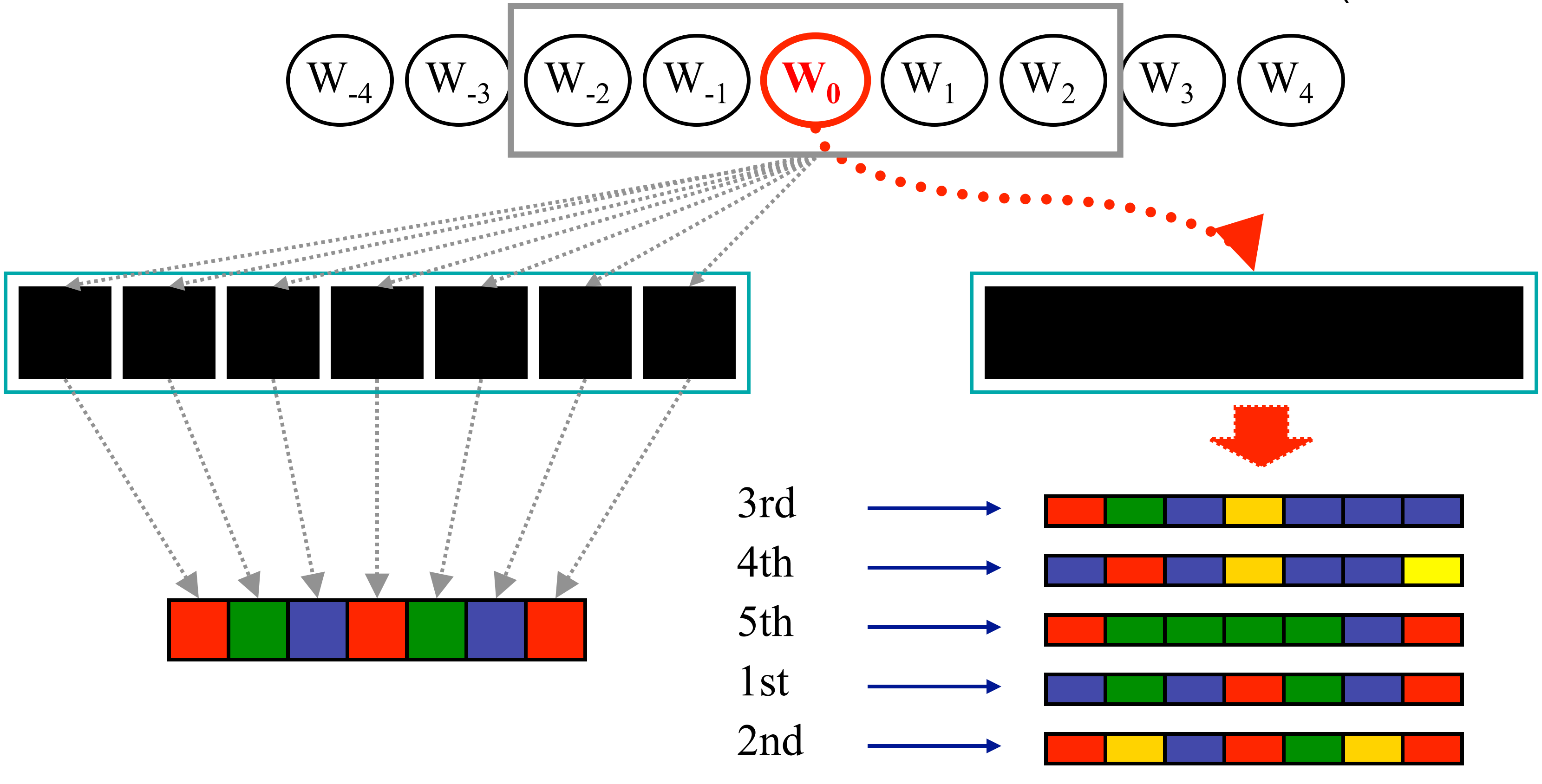
VB	VBD	VBG	VBN	VBP	VBZ
go	went	going	gone	go	goes

- Same as "POS Tagging" in English

Morphological Disambiguation *in Arabic*

- Morphological tag has 14 subtags corresponding to different linguistic categories
  - Example:Verb
    Gender(2), Number(3), Person(3), Aspect(3), Mood(3), Voice(2), Pronominal clitic(12), Conjunction clitic(3)
- 22,400 possible tags
  - Different possible subsets
- 2,200 appear in Penn Arabic Tree Bank Part 1 (140K words)
- Example solution: MADA (Habash&Rambow 2005)

MADA (Habash&Rambow 2005;Roth et al. 2008)
MADAMIRA (Pasha et al., 2014)



MORPHOLOGICAL CLASSIFIERS

- Multiple independent classifiers
- Corpus-trained

RANKER

- Heuristic or corpus-trained

MORPHOLOGICAL ANALYZER

- Rule-based
- Human-created

MADA 3.2 (MSA) Evaluation

Accuracy	PATB 3 Blind Test		
	Baseline	**MADA**	**Error ↓**
All	**74.8%**	**84.3%**	**38%**
POS + Features	**76.0%**	**85.4%**	**39%**
All Diacritics	**76.8%**	**86.4%**	**41%**
Lemmas	**90.4%**	**96.1%**	**60%**
Partial Diacritics	**90.6%**	**95.3%**	**50%**
Base POS	**91.1%**	**96.1%**	**56%**
Segmentation	**96.1%**	**99.1%**	**77%**

Baseline: most common analysis per word in training

wkAtb وكاتب

and (the) writer of

wakAtibu

kAtib_1

pos:noun

prc3:0 prc2:wa_conj prc1:0 prc0:0 per:3 asp:na vox:na mod:na gen:m num:s stt:c cas:n enc0:0

w+ kAtb

MADAMIRA DEMO

- http://nlp.ldeo.columbia.edu/madamira/

Next Time

- Read J+M Chap 12
- Assignment #3 will be given out