Statistical-based POSTagging

Corpus Annotation

- A corpus (plural corpora) is a computer-readable collection of NL text (or speech) used as a source of information about the language: e.g. what words/constructions can occur in practice, and with what frequencies.
- The usefulness of a corpus can be enhanced by annotating each word with a POS tag, e.g.
  - Our/PRP\$ enemies/NNS are/VBP innovative/JJ and/CC resourceful/JJ ,/, and/CC so/RB are/VB we/PRP ./. They/PRP never/RB stop/VB thinking/VBG about/IN new/JJ ways/NNS to/TO harm/VB our/PRP\$ country/NN and/CC our/PRP\$ people/NN, and/CC neither/DT do/VB we/PRP ./.
- Typically done by an automatic tagger, then hand-corrected by a native speaker, in accordance with specified tagging guidelines.

Sources of information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VB(D) IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb….
- The latter proves the most useful, but the former also helps

More and Better Features ➔ Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word the: the → DT
  - Lowercased word Importantly: importantly → RB
  - Prefixes unfathomable: un- → JJ
  - Suffixes Importantly: -ly → RB
  - Capitalization Meridian: CAP → NNP
  - Word shapes 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
  - Maxent P(t|w): 93.7% overall / 82.6% unknown

Overview: POS Tagging Accuracies

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - MEMM tagger: 96.9% / 86.9%
  - Bidirectional dependencies: 97.2% / 90.0%
  - Upper bound: ~98% (human agreement)

Most errors on unknown words

How to improve supervised results?

- Build better features!

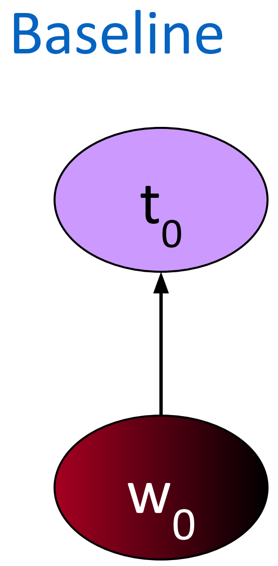
		RB				
PRP	VBD	IN	RB IN	PRP	VBD	.
They	left	as soon as		he	arrived	.

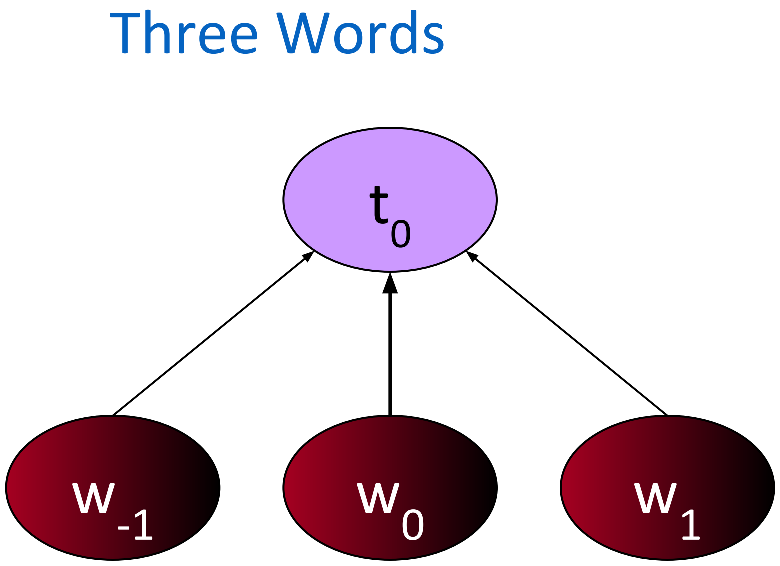
  - We could fix this with a feature that looked at the next word

JJ				
NNP	NNS	VBD	VBN	.
Intrinsic	flaws	remained	undetected	.

  - We could fix this by linking capitalized words to their lowercase versions

Tagging Without Sequence Information





Model	Features	Token	Unknown	Sentence
Baseline	56,805	**93.69%**	82.61%	26.74%
3Words	239,767	**96.57%**	86.78%	48.27%

Using words only in a straight classifier works as well as a basic (HMM or discriminative) sequence model!!

Beyond Classification Learning

- Standard classification problem assumes individual cases are disconnected and independent (i.i.d.: independently and identically distributed).
- Many NLP problems do not satisfy this assumption and involve making many connected decisions, each resolving a different ambiguity, but which are mutually dependent.
- More sophisticated learning and inference techniques are needed to handle such situations in general.

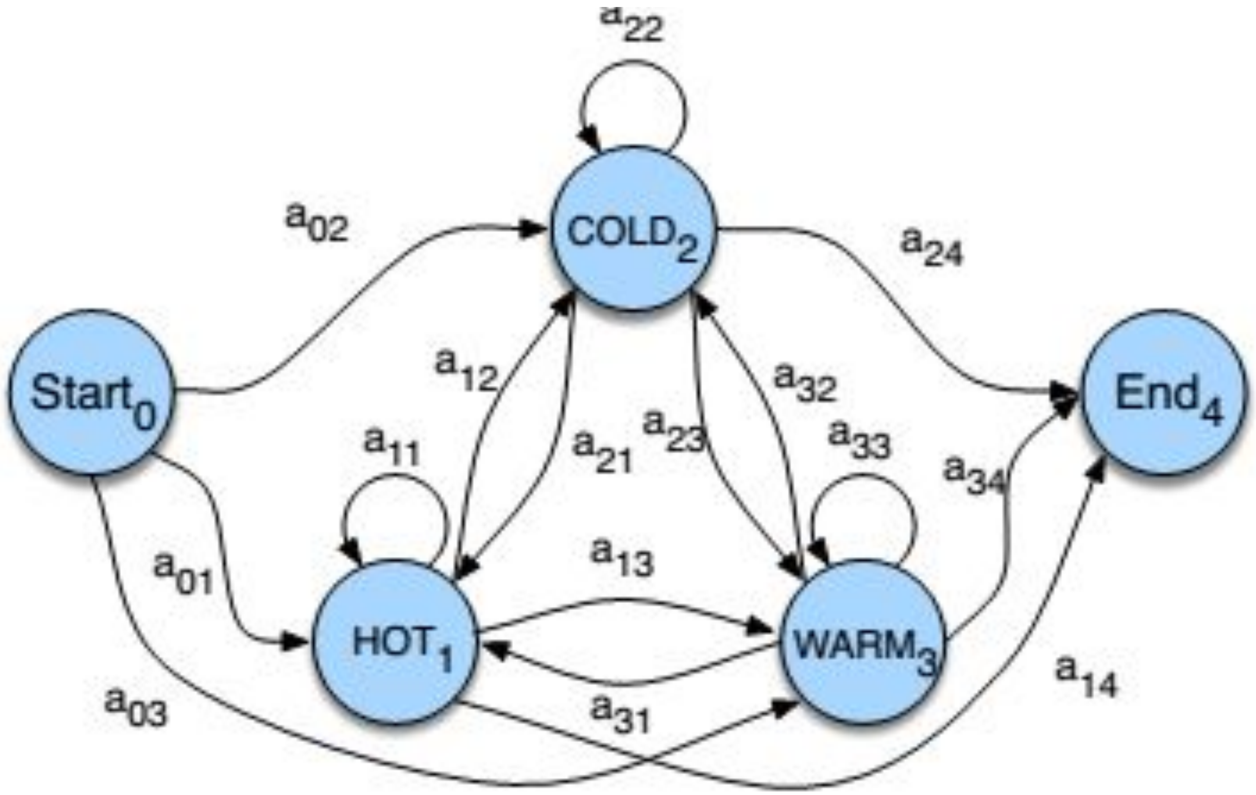
HMM Model

- The HMM is a **sequence model**.
- A sequence model or sequence classifier is a model whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of **observations** to a sequence of **labels**.
- An HMM is a **probabilistic sequence model**: given a sequence of units (words, letters, morphemes, sentences, whatever), they compute a probability distribution over possible sequences of labels and choose the best label sequence.
- **Sequence labeling task : PoS Tagging, NER, Speech Recognition**

Markov Model / Markov Chain

- A finite state machine with probabilistic state transitions.
- Makes Markov assumption that next state only depends on the current state and independent of previous history.

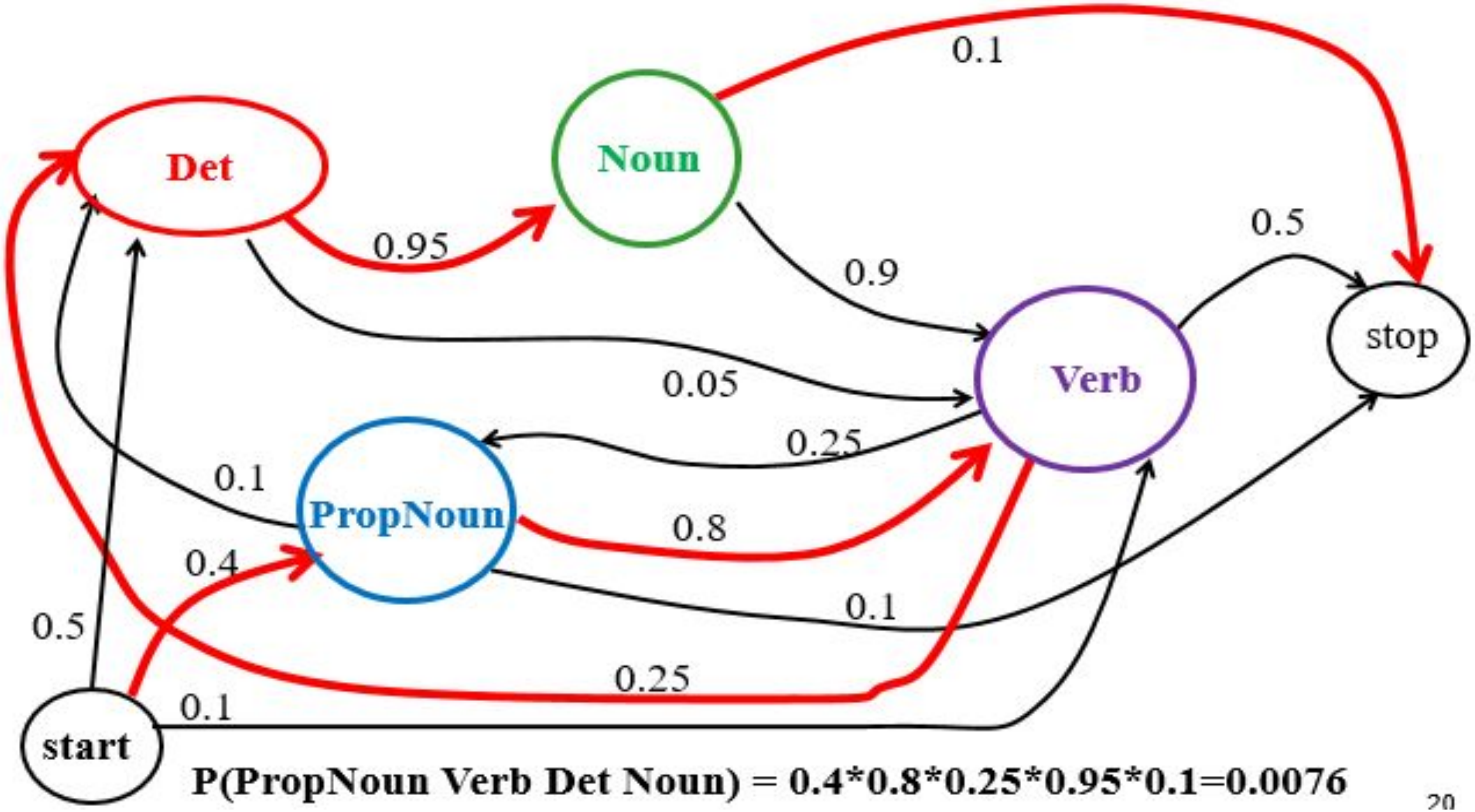
Markov Chains



Source : jurafsky

- Markov chain is an extension of Finite Automata, especially weighted finite automaton.
- Markov chain is a special case which the weights are probability so that it sums to 1, and not ambiguous

Sample Markov Model for POS

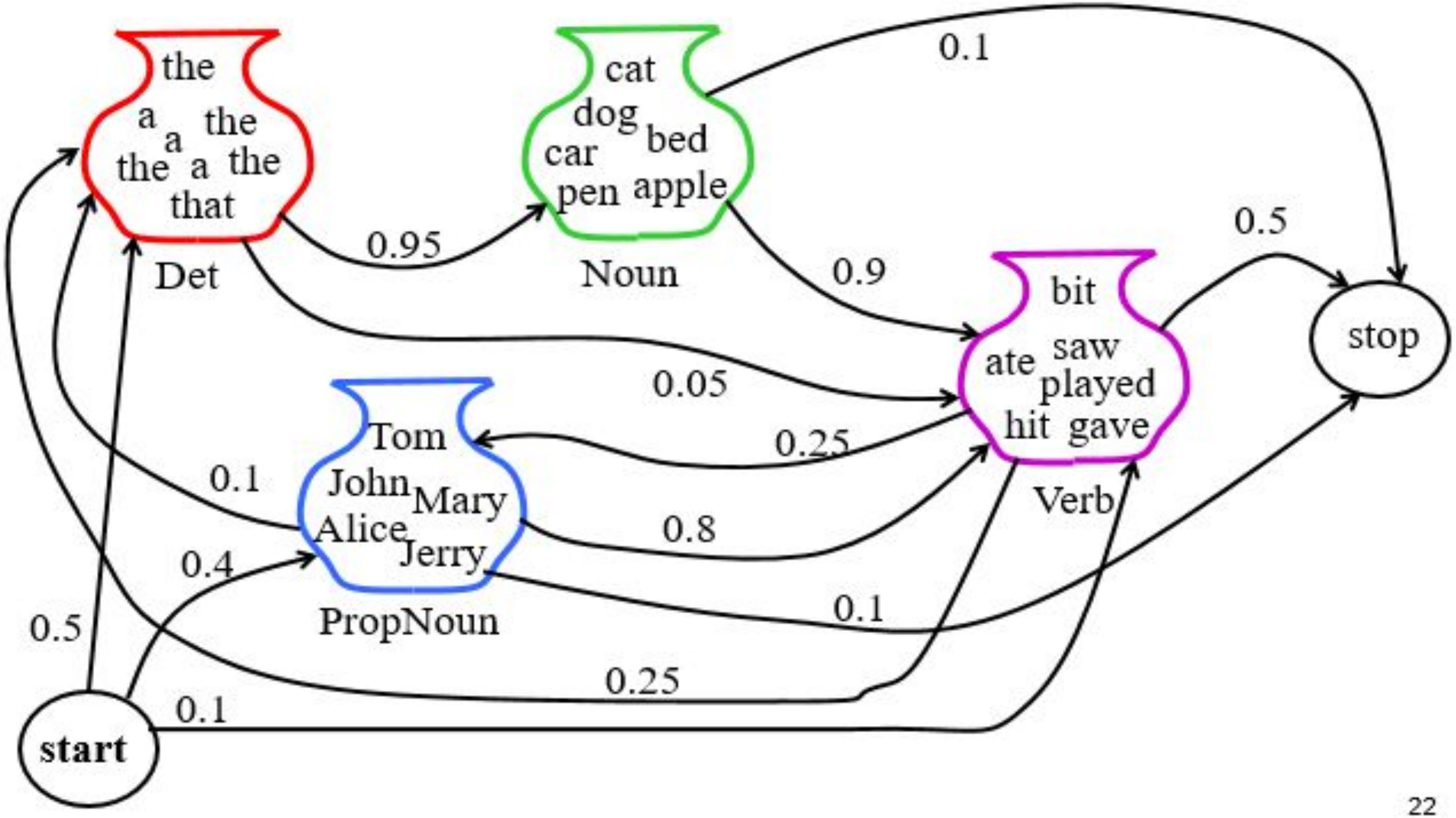


P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1=0.0076

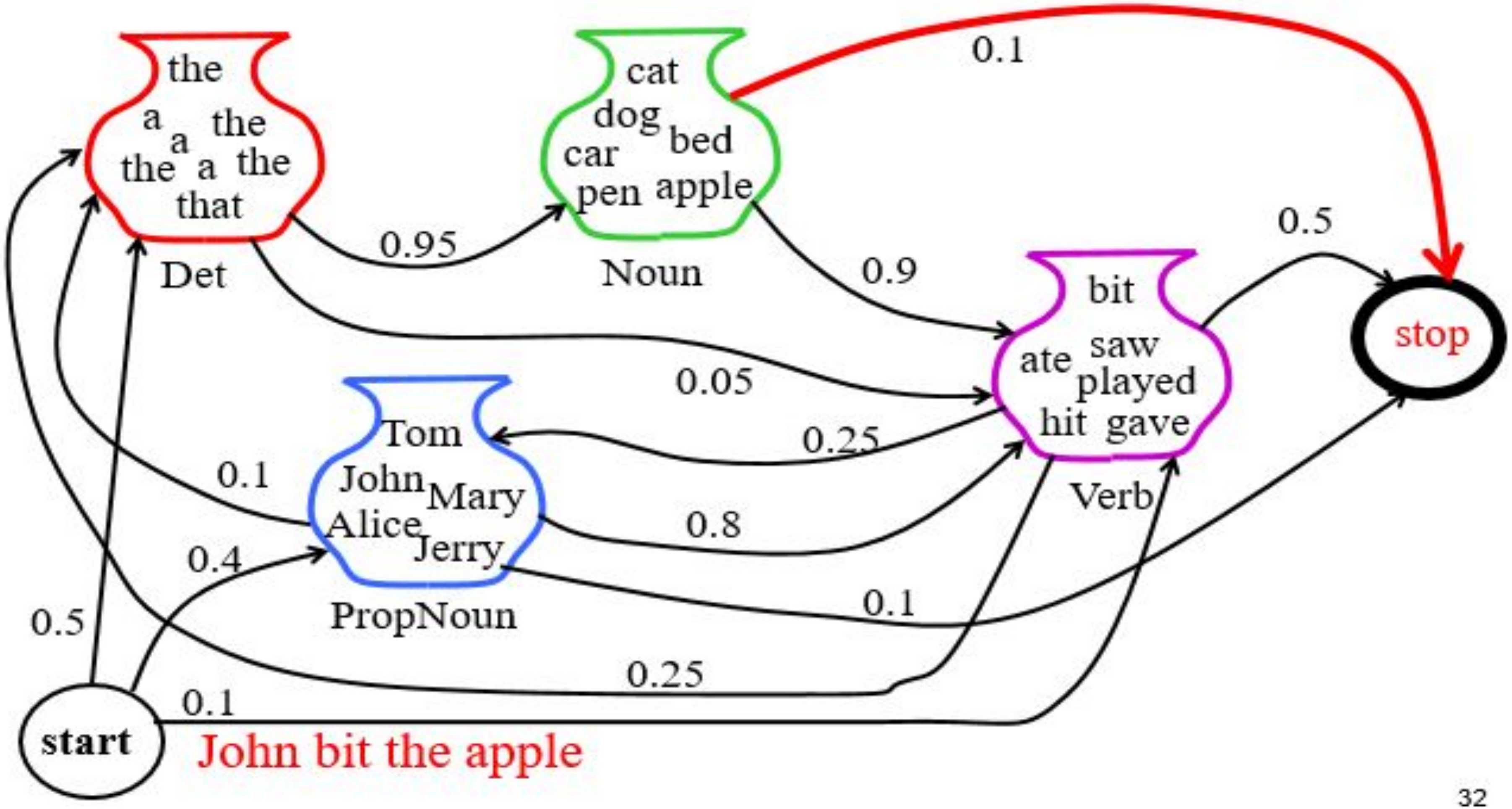
Hidden Markov Model

- Probabilistic generative model for sequences.
- Assume an underlying set of ***hidden*** (unobserved, latent) states in which the model can be (e.g. parts of speech).
- Assume probabilistic transitions between states over time (e.g. transition from POS to another POS as sequence is generated).
- Assume a ***probabilistic*** generation of tokens from states (e.g. words generated for each POS).

Sample HMM for POS



Sample HMM Generation



Formal Definition of an HMM

- A set of *N +2* states $S=\{s_0,s_1,s_2, \ldots s_N, s_F\}$
  - Distinguished start state: s_0
  - Distinguished final state: s_F
- A set of *M* possible observations $V=\{v_1,v_2\ldots v_M\}$
- A state transition probability distribution $A=\{a_{ij}\}$

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i) \qquad 1 \leq i, j \leq N \text{ and } i = 0, j = F$$

$$\sum_{j=1}^{N} a_{ij} + a_{iF} = 1 \quad 0 \leq i \leq N$$

- Observation probability distribution for each state *j* $B=\{b_j(k)\}$

$$b_j(k) = P(X_t = v_k \mid q_t = s_j) \quad 1 \leq j \leq N \quad 1 \leq \mathrm{k} \leq \mathrm{M}$$

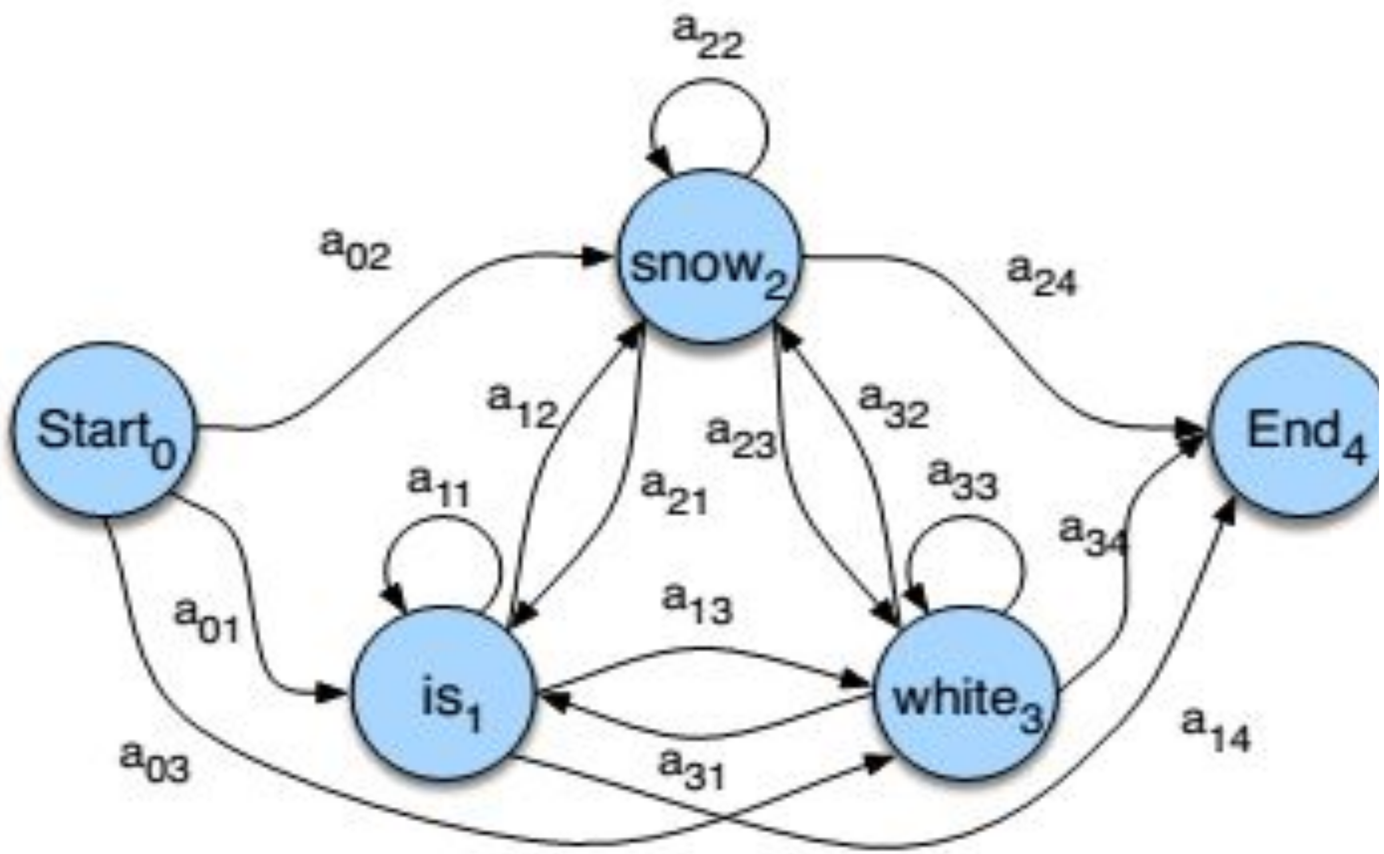
- Total parameter set $\lambda=\{A,B\}$

HMM Generation Procedure

- To generate a sequence of T observations: $O = o_1\ o_2\ \ldots\ o_T$

Set initial state $q_1 = s_0$
For t = 1 to T
    Transit to another state $q_{t+1} = s_j$ based on transition distribution a_{ij} for state q_t
    Pick an observation $o_t = v_k$ based on being in state q_t using distribution $b_{qt}(k)$

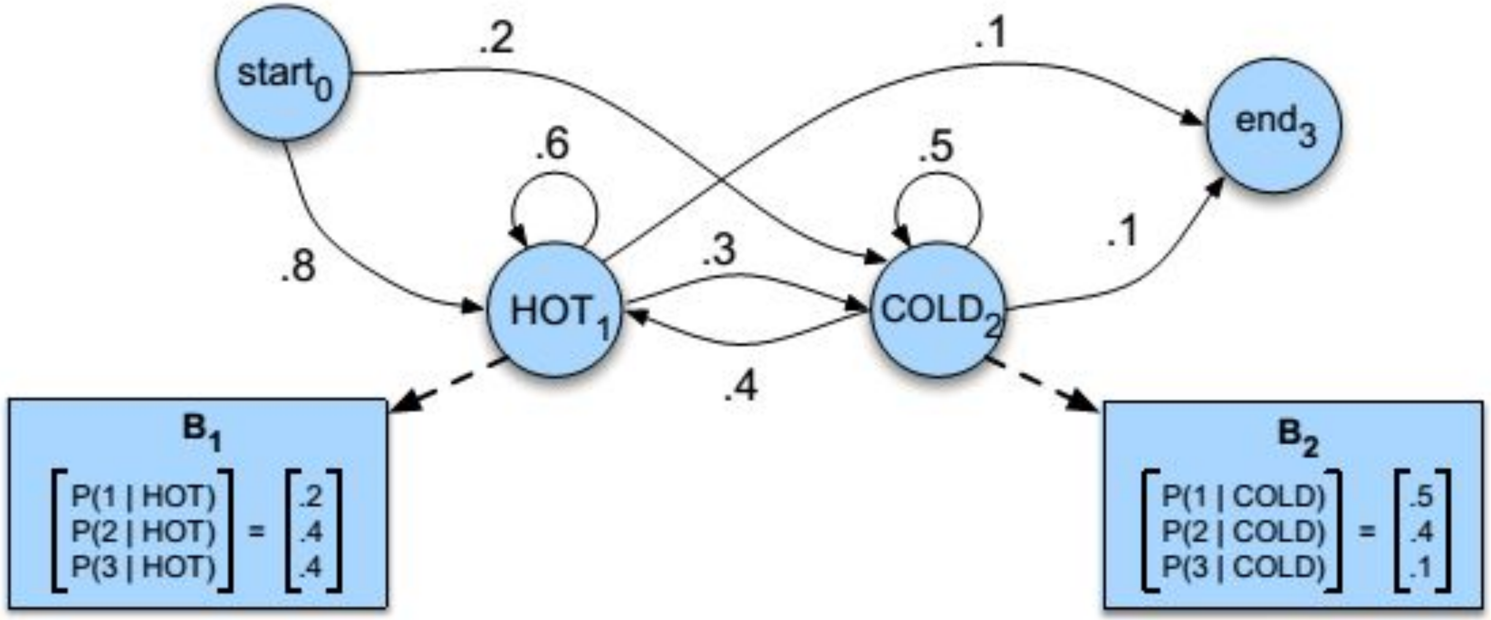
Markov Chain



Source : jurafsky

- This markov chain represent bigram language model. Can you see that?

The Hidden Markov Model



- Given how many **Ice Cream[observation]** Jason Eisner eats everyday in summer, figure out the **weather status[hidden]** each day

Source : jurafsky

HMM Components

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$
$QA = \{q_x, q_y ...\}$	a set $QA \subset Q$ of legal **accepting states**

Some Probabilities

- We want to find : $q_1^n = \underset{q_1^n}{\operatorname{argmax}} P(q_1^n | o_1^n)$
- Using Bayes' rule : $q_1^n = \underset{q_1^n}{\operatorname{argmax}} \frac{P(o_1^n | q_1^n) P(q_1^n)}{P(o_1^n)}$
- Drop denominator (why?) : $q_1^n = \underset{q_1^n}{\operatorname{argmax}} P(o_1^n | q_1^n) P(q_1^n)$

Assumptions

- $q_1^n = \underset{q_1^n}{\operatorname{argmax}} P(o_1^n | q_1^n) P(q_1^n)$

There are 2 assumptions in HMM :

1. 1st order Markov Assumption : probability of a particular state depends only on the previous state
$$\boldsymbol{P(q_i | q_1, q_2, \dots, q_{i-1}) = P(q_i | q_{i-1})}$$
2. The probability of an output observation o_i depends only on the state that produce the observation which is q_i
$$\boldsymbol{P(o_i | q_1, \dots, q_i, \dots, q_N, o_1, \dots, o_i, \dots, o_N) = P(o_i | q_i)}$$

Problems related to HMM

1. Likelihood : Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$
2. Decoding : Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the best hidden state sequence Q.
3. Learning : Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B

HMM for PoS Tagging

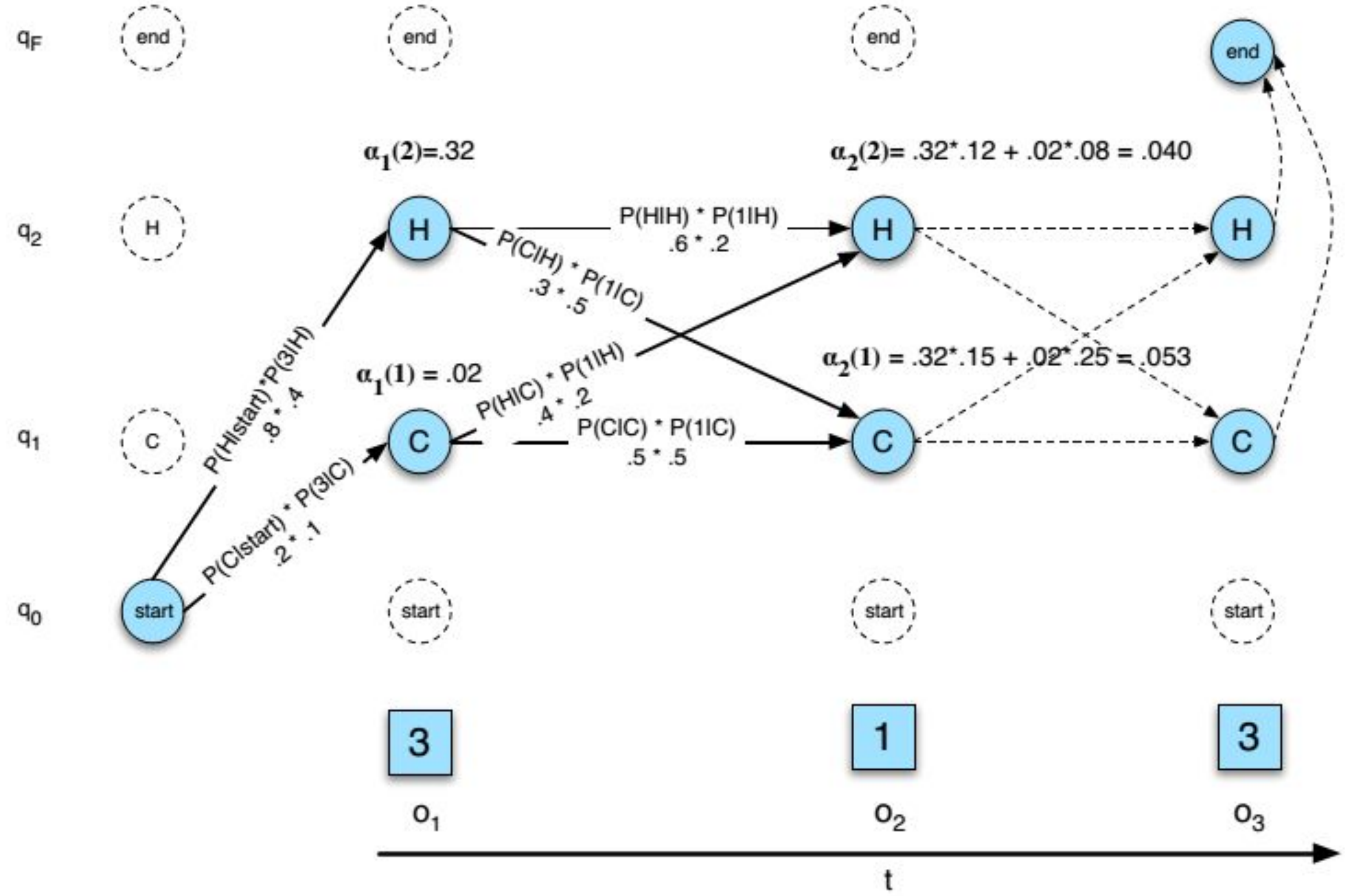
- From : Janet will back the bill → **OBSERVED**
- To : NNP MD VB DT NN → **HIDDEN**

- **Which problem is this ?**

Likelihood

- Ex : what is the likelihood of eating ice cream with a sequence of 3 1 3 ?
- P(3 1 3)= P(3 1 3, cold cold cold)+P(3 1 3, cold cold hot)+…. P(3 1 3, hot hot hot)

Likelihood

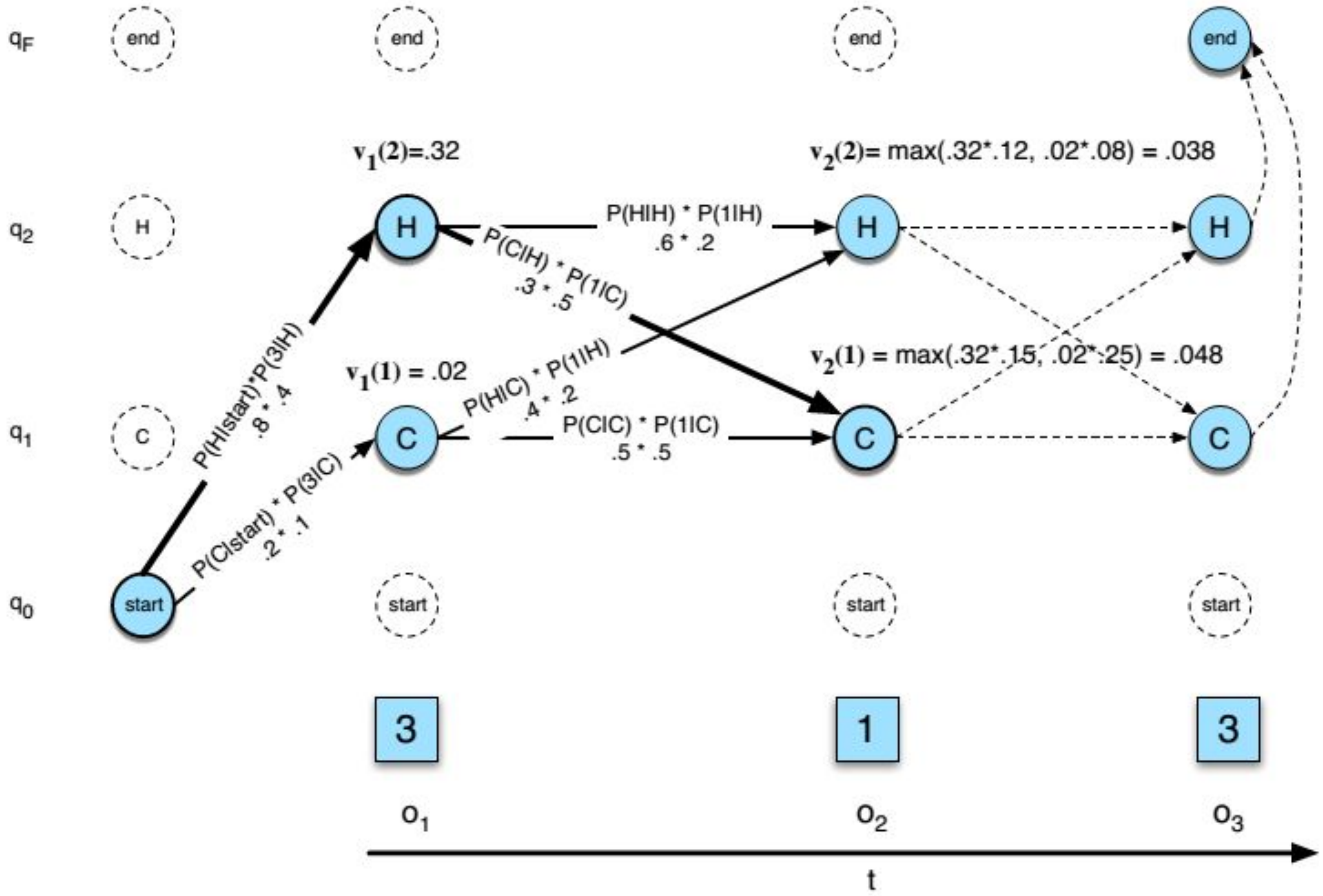


q_F
q_2
q_1
q_0
end
end
end
end
$\alpha_1(2)$=.32
$\alpha_2(2)$= .32*.12 + .02*.08 = .040
H
H
H
H
P(H|H) * P(1|H)
.6 * .2
P(C|H) * P(1|C)
.3 * .5
P(H|start)*P(3|H)
.8 * .4
$\alpha_1(1)$ = .02
P(H|C) * P(1|H)
.4 * .2
$\alpha_2(1)$ = .32*.15 + .02*.25 = .053
C
C
C
C
P(C|C) * P(1|C)
.5 * .5
P(C|start) * P(3|C)
.2 * .1
start
start
start
start
3
1
3
o_1
o_2
o_3
t

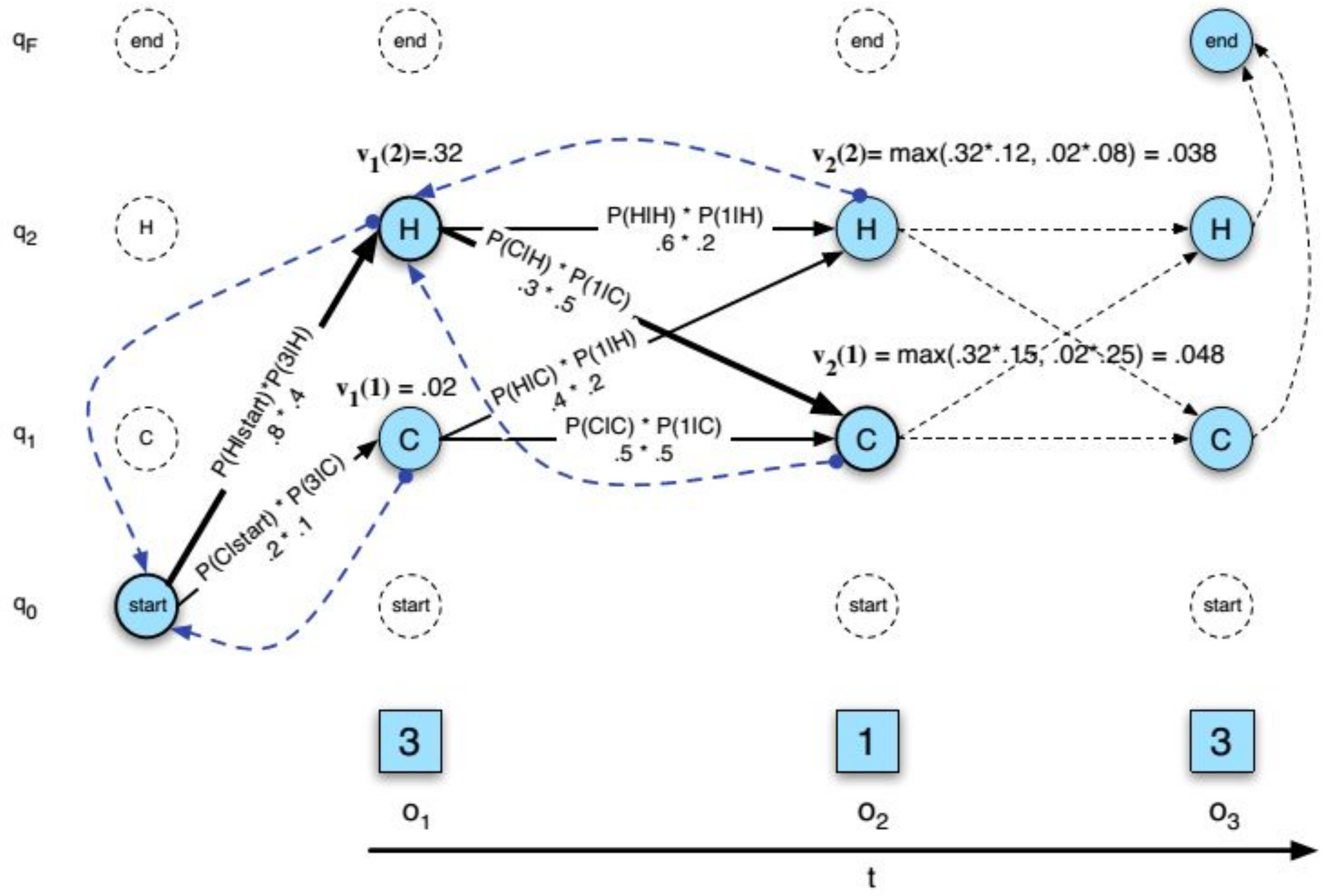
Decoding

- Finding **the best** hidden states given observations
- Ex : What is the best sequence of weather given ice cream observation of 3 1 3 ?
- Approach :
  - Brute force : 3 1 3, Find likelihood (problem 1) of all possible states combination with length of 3, ex : C C C, C C H, …, H H H, then choose sequence that give the maximum likelihood
  - **Viterbi Algorithm**
    - A kind of dynamic programming

Decoding : Viterbi



Viterbi Backtrace



PoS Tagging

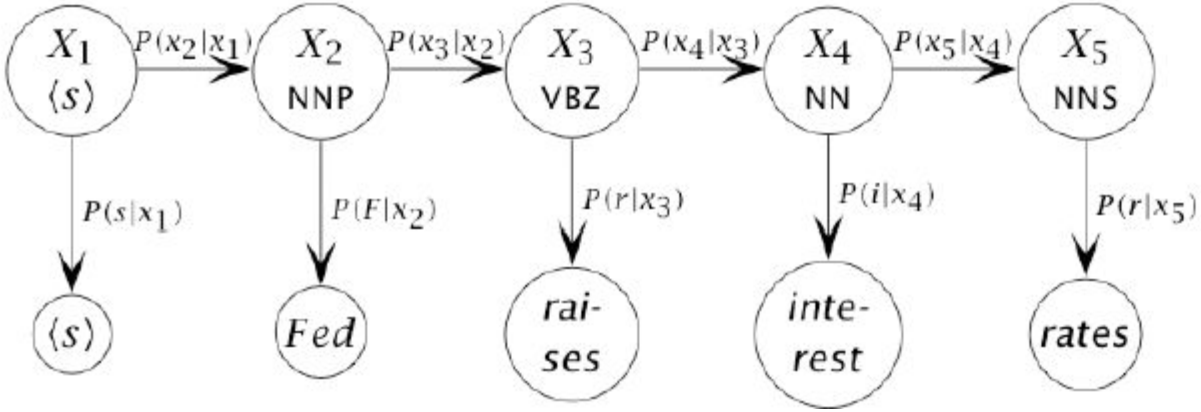
- earnings growth took a **back/JJ** seat
- a small building in the **back/NN**
- a clear majority of senators **back/VBP** the bill
- Dave began to **back/VB** toward the door
- enable the country to buy **back/RP** about debt
- I was twenty-one **back/RB** then

- How to tag a word correctly ?
    1. Look at the word
    2. Look at the previous tag ?

- Janet will back the bill
- Janet/NNP will/MD back/VB the/DT bill/NN

Model HMM

- HMM adalah sebuah model sekuens, yaitu model yang akan memberikan label/kategori untuk tiap unit dalam sebuah sekuens. Kata adalah unit dalam sebuah kalimat.
- HMM adalah sebuah probabilistic sequence model. Cara kerja HMM berdasar distribusi probability semua kemungkinan label untuk sebuah sekuens, dan kemudian dipilih yang paling baik.
- HMM sebagai sebuah Bayesian Network



Baris atas: unobserved states (POSTag)

Baris bawah: observed output/observation (kata)

HMM, Bayes Rule, dan Probabilities (1)

- Tujuan HMM

  Memperkirakan sekuens tag terbaik

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

- Penggunaan Bayes Rule

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

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

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

HMM, Bayes Rule, dan Probabilities (2)

- Dua jenis probabilities:
  - Tag transition probabilities $p(t_i|t_{i-1})$

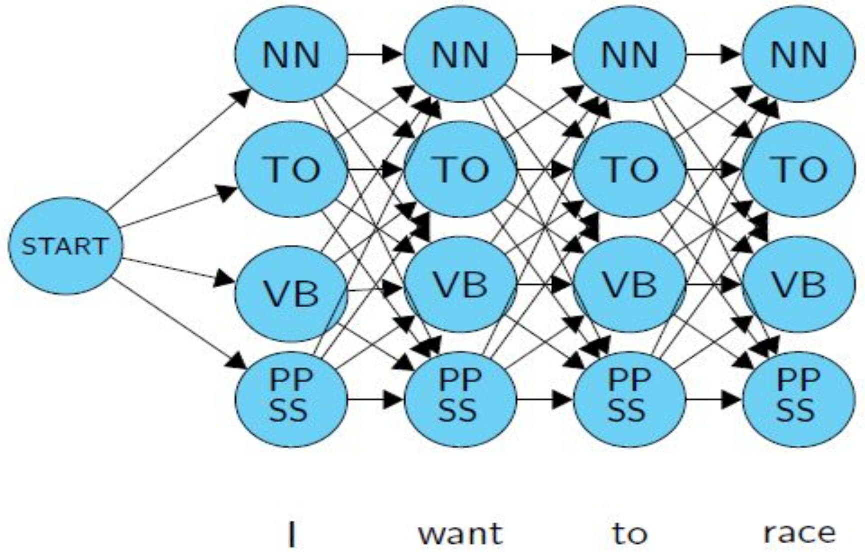
$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

  - Word likelihood probabilities $p(w_i|t_i)$

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

Algoritma Viterbi untuk POSTagging (1)

- POSTagging: *decoding,* salah satu dari 3 persoalan utama yang dapat diselesaikan dengan model HMM. *Diberikan sebuah sekuens observasi O and sebuah model HMM λ= (A,B), temukan sekuens hidden state terbaik Q* .
- Algoritma Viterbi: menggunakan metode *dynamic programming* untuk mendapat sekuens POSTag terbaik.
- Contoh ilustrasi sebuah trellis HMM:



Statistical POS tagging

- What is the most likely sequence of tags for the given sequence of words w

$$
\begin{aligned}
\operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}) &= \operatorname*{argmax}_{\mathbf{t}} \frac{P(\mathbf{t},\mathbf{w})}{P(\mathbf{w})} \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t},\mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{w}|\mathbf{t})
\end{aligned}
$$

P(DT JJ NN | a smart dog) = P(DD JJ NN a smart dog) / P (a smart dog)
∝ P(DD JJ NN a smart dog)
= P(DD JJ NN) P(a smart dog | DD JJ NN)

How you predict the tags?

- Two types of information are useful
  - Relations between words and tags
  - Relations between tags and tags
    - DT NN, DT JJ NN…

Transition Probability

- Joint probability $P(\boldsymbol{t}, \boldsymbol{w}) = P(\boldsymbol{t})P(\boldsymbol{w}|\boldsymbol{t})$
- $P(\boldsymbol{t}) = P(t_1, t_2, \dots t_n)$

$$= P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_2, t_1) \dots P(t_n \mid t_1 \dots t_{n-1})$$

$$\sim \mathrm{P}(\mathrm{t}_1)\mathrm{P}(t_2 \mid t_1)P(t_3 \mid t_2) \dots P(t_n \mid t_{n-1})$$

$$= \Pi_{i=1}^{n} P(t_i \mid t_{i-1})$$

Markov assumption

- Bigram model over POS tags!
(similarly, we can define a n-gram model over POS tags, usually we called high-order HMM)

Emission Probability

- Joint probability $P(\boldsymbol{t}, \boldsymbol{w}) = P(\boldsymbol{t})P(\boldsymbol{w}|\boldsymbol{t})$
- Assume words only depend on their POS-tag
- $P(\boldsymbol{w}|\boldsymbol{t}) \sim P(w_1 \mid t_1)P(w_2 \mid t_2) \ldots P(w_n \mid t_n)$

$$= \Pi_{i=1}^{n} P(w_i|t_i)$$

Independent assumption

i.e., P(a smart dog | DD JJ NN)

= P(a | DD) P(smart | JJ) P(dog | NN)

Put them together

- Joint probability $P(\boldsymbol{t}, \boldsymbol{w}) = P(\boldsymbol{t})P(\boldsymbol{w}|\boldsymbol{t})$
- $P(\boldsymbol{t}, \boldsymbol{w})$

$$= \mathrm{P}(\mathrm{t}_1)\mathrm{P}(\mathrm{t}_2 \mid t_1)P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})$$
$$P(w_1 \mid t_1)P(w_2 \mid t_2) \ldots P(w_n \mid t_n)$$
$$= \Pi_{i=1}^{n} P(w_i|t_i)P(t_i \mid t_{i-1})$$

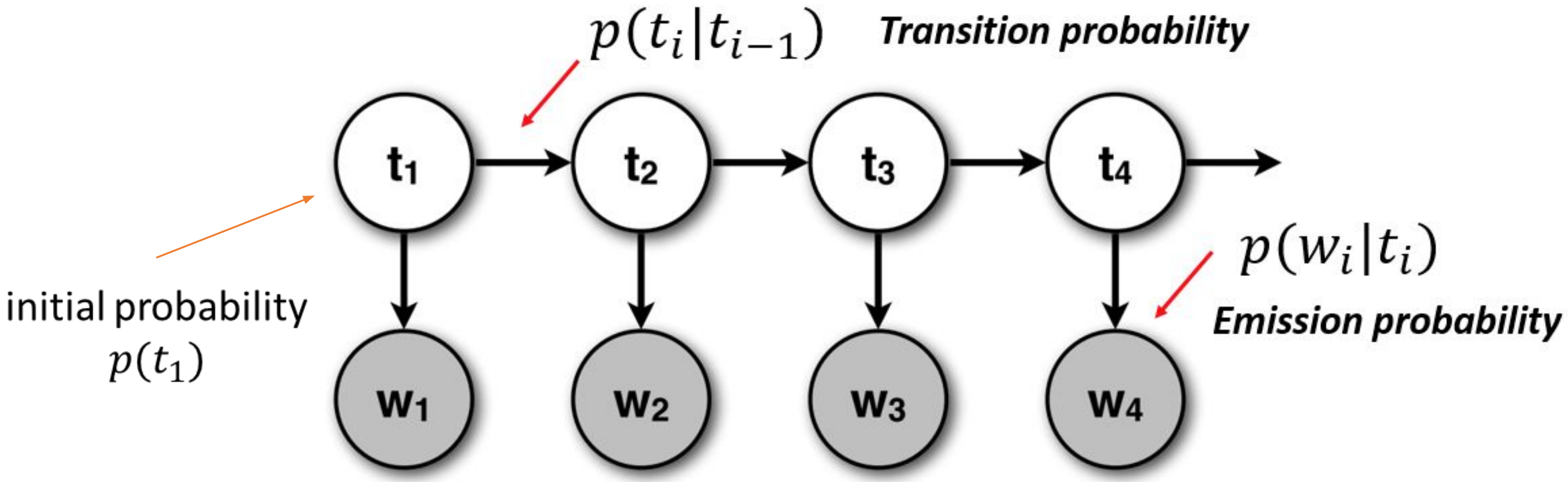
e.g., P(a smart dog , DD JJ NN)

= P(a | DD) P(smart | JJ) P(dog | NN)

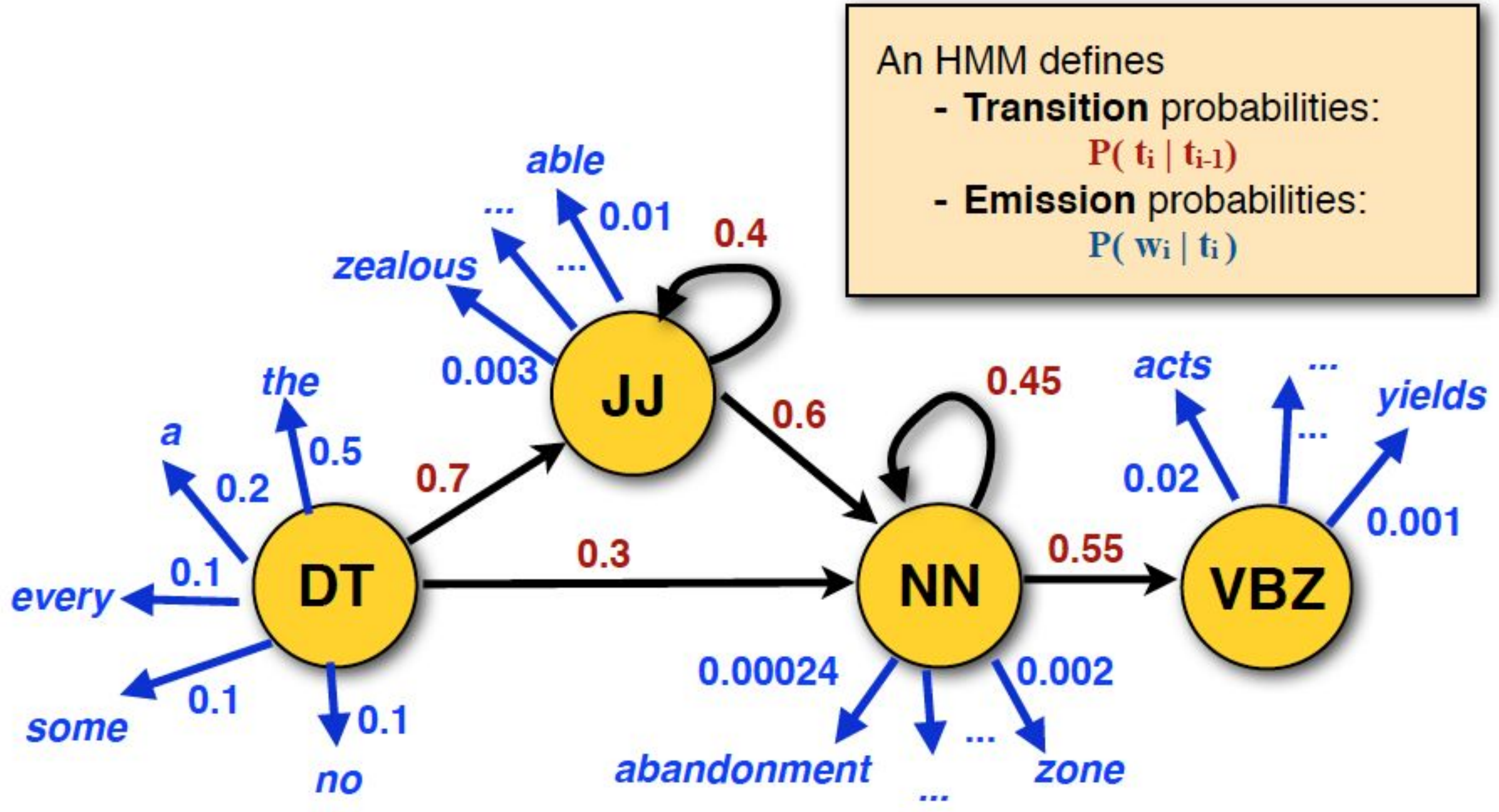
P(DD | start) P(JJ | DD) P(NN | JJ)

Put them together

- Two independent assumptions
  - Approximate P(**t**) by a bi(or N)-gram model
  - Assume each word depends only on its POStag



HMMs as probabilistic FSA



Julia Hockenmaier: Intro to NLP

Table representation

Transition Matrix A

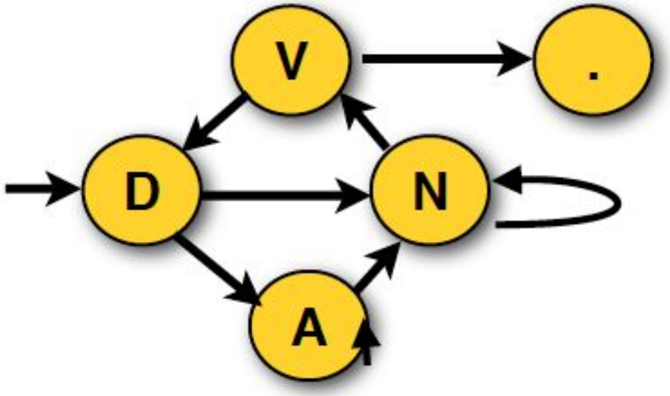
	D	N	V	A	.
D		0.8		0.2	
N		0.7	0.3		
V	0.6				0.4
A		0.8		0.2	
.					

Emission Matrix B

	the	*man*	*ball*	*throws*	*sees*	*red*	*blue*	.
D	1.0							
N		0.7	0.3					
V				0.6	0.4			
A						0.8	0.2	
.								1

Initial state vector π

	D	N	V	A	.
π	1.0				



Let $\lambda = \{A, B, \pi\}$ represents all parameters

Algoritma Viterbi untuk POSTagging (2)

- Buat matriks probabilities, berdasar transition probabilities dan emission probabilities

q_4	NN	0				
q_3	TO	0				
q_2	VB	0				
q_1	PPSS	0				
q_o	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- Isi tiap cell, kolom per kolom dari kata paling kiri.

$$v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$$

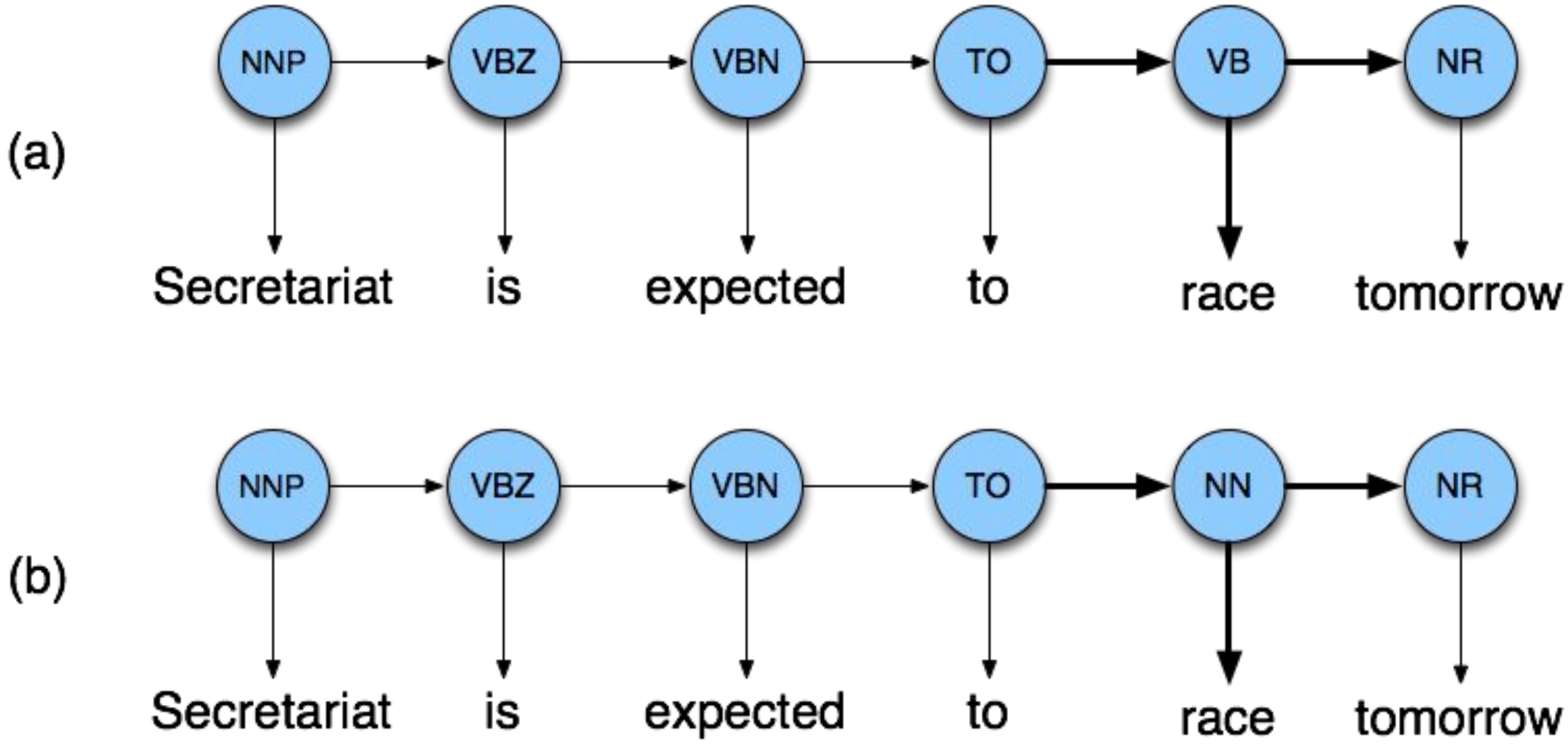
vi-1(k) adalah previous Viterbi path probability, akj: transition probability, dan bj(wi): emission probability

- Sampai di kata paling akhir, cek nilai *probability* tertinggi, dan lakukan *backtrace*.

Toward a Bigram-HMM Tagger

- $\text{argmax}_T\ P(T|W)$
- $\text{argmax}_T P(T)P(W|T)$
- $\text{argmax}_t P(t_1 \ldots t_n)P(w_1 \ldots w_n|t_1 \ldots t_n)$
- $\text{argmax}_t [P(t_1)P(t_2|t_1) \ldots P(t_n|t_{n-1})][P(w_1|t_1)P(w_2|t_2) \ldots P(w_n|t_n)]$
- To tag a single word: $t_i = \text{argmax}_j\ P(t_j|t_{i-1})P(w_i|t_j)$
- How do we compute $P(t_i|t_{i-1})$?
  - $c(t_{i-1}t_i)/c(t_{i-1})$
- How do we compute $P(w_i|t_i)$?
  - $c(w_i,t_i)/c(t_i)$
- How do we compute the most probable tag sequence?
  - Viterbi

Disambiguating “race”



HMM Tagger

- Intuition: Pick the most likely tag for this word.
- Let $T = t_1,t_2,\ldots,t_n$
  Let $W = w_1,w_2,\ldots,w_n$
- Find POS tags that generate a sequence of words, i.e., look for most probable sequence of tags T underlying the observed words W.

Example

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

Latihan HMM

- Diberikan data training sebagai berikut :
- \- saya/A ingin/B makan/F ikan/D
  \- saya/A makan/C seafood/D kemaren/G
  \- dia/A ingin/B tidur/F nyenyak/E
  \- semalam/G saya/A tidur/C
-
- Hitung likelihood dari sequence berikut ini :
- dia/A ingin/B makan/C ikan/D kemaren/G