Natural Language Processing

Some screenshots are taken from NLP course by Jufrasky
— Used only for educational purpose

LNMIIT

Sequence Modeling

Sequence Modelling

- English Word Classes and Tag Sets
- POS-Tagging
- Rule-based approach to POS- Tagging
- HMM POS-Tagging
- Hidden Markov Model
- Forward algorithm and Viterbi Algorithm

English Word and Classes

- Thrax of Alexandria (100 BCE) was the first two write grammatical sketch for Greek
- Defined syntax, diphthong, clitic and analogy
- Noun, verb, pronoun, preposition, adverb, conjunction, participle and article
- Basis for all European language

Part-of-Speech Tagging

- Many names for PoS Tagging: Lexical categories, Word classes, Morphological classes, Lexical tags and so on…
- Labelling with PoS gives more information about a word and its neighbours
    - Not only verbs versus nouns but more — like about possessive pronouns and personal pronouns
    - Possessive pronouns — my, your, our her, his
    - Personal pronouns — I, you, he, she

Part-of-Speech Tagging

- Penn Treebank — 45 word classes (Marcus et al., 1993)
- C5 tagset — 61 word classes (Lanchaster UCREL Project 1997)
- Brown Corpus — 87 word classes (Francis et al., 1979 and 1982)
- C7 tagset — 146 word classes (Garside et al., 1997)

Part-of-Speech Tagging

- Uses of PoS:
  - Parsing
  - Information Extraction
  - Machine Translation

Methods to do PoS Tagging

- Rule-based tagging
- Statistical methods (HMM tagging and Maximum Entropy Tagging)
- Transformation-based tagging
- Memory-based tagging

LNMIIT

English Word Classes

Closed Class

No new additions
For example: Prepositions

Open Class

nouns

verbs

adjectives

adverbs

new nouns, adjectives, verbs and adverbs are added regularly

English Word Classes

- Nouns:
  - Proper nouns — names of a person, company name and so on (capitalised and not proceeded by an article)
  - Common nouns — This if two types
    - Count nouns — Singlular / Plural — (Eg. goat / goats)
    - Mass nouns — (Eg. Snow, Salt)

English Word Classes

- Verb — action or processes
  - Eg. draw, provide, differ, go
  - Morphological forms - Eg. eat, eats, eating, eaten
- Auxillary Verbs — discussed in closed forms

LNMIIT

English Word Classes

- Adjectives — describes properties or qualities of a noun
  - Eg. white, black, old, young, good, bad
  - There are languages with no adjectives — Korean

English Word Classes

- Adverbs — Modifying verbs or adverbs
  - Eg. *Unfortunately, John walked home extremely slowly yesterday*
  - Four types — directional, degree, manner, temporal

English Word Classes

- Closed Class — differ with respect to languages
  - Prepositions — on, in, under, ovber, by, at
  - Determiners — a, an, the, this that
  - Pronouns — he, she, who, I
  - Conjunctions — and, but, or
  - Auxillary verbs — can, may should, are
  - Particles — up, down, on, off, in, out
  - Numerals — one, two, three, first second

English Word Classes

- Prepositions — that occur before nouns
- Particle — used in combination with a verb
  - Sometimes verb and auricle act as a single semantic unit
  - Above point gives a very different meaning
  - For Eg. *turn down* — reject, *rule out* — eliminate, *find out* — discover
  - Extremely difficult to classify it is a preposition or particle

Prepositions from CELEX

of	540,085	through	14,964	worth	1,563	pace	12
in	331,235	after	13,670	toward	1,390	nigh	9
for	142,421	between	13,275	plus	750	re	4
to	125,691	under	9,525	till	686	mid	3
with	124,965	per	6,515	amongst	525	o'er	2
on	109,129	among	5,090	via	351	but	0
at	100,169	within	5,030	amid	222	ere	0
by	77,794	towards	4,700	underneath	164	less	0
from	74,843	above	3,056	versus	113	midst	0
about	38,428	near	2,026	amidst	67	o'	0
than	20,210	off	1,695	sans	20	thru	0
over	18,071	past	1,575	circa	14	vice	0

CELEX is from Linguistic Data Consortium

Particles

aboard	aside	besides	forward(s)	opposite	through
about	astray	between	home	out	throughout
above	away	beyond	in	outside	together
across	back	by	inside	over	under
ahead	before	close	instead	overhead	underneath
alongside	behind	down	near	past	up
apart	below	east, etc.	off	round	within
around	beneath	eastward(s),etc.	on	since	without

English Word Classes

- Determiners: comes with noun, articles included
- Conjunction: Doing two phrases / sentences
    - Coordinating injunctions — *and, or, but*
    - Subordinating injunctions — *I thought that you might come*
- *Pronouns:*
    - *Personal pronouns — you, she, I*
    - *Possesive pronouns — my, your, his*
    - *Wh-pronouns — what, who, whom, whoever*

Conjunctions

and	514,946	yet	5,040	considering	174	forasmuch as	0
that	134,773	since	4,843	lest	131	however	0
but	96,889	where	3,952	albeit	104	immediately	0
or	76,563	nor	3,078	providing	96	in as far as	0
as	54,608	once	2,826	whereupon	85	in so far as	0
if	53,917	unless	2,205	seeing	63	inasmuch as	0
when	37,975	why	1,333	directly	26	insomuch as	0
because	23,626	now	1,290	ere	12	insomuch that	0
so	12,933	neither	1,120	notwithstanding	3	like	0
before	10,720	whenever	913	according as	0	neither nor	0
though	10,329	whereas	867	as if	0	now that	0
than	9,511	except	864	as long as	0	only	0
while	8,144	till	686	as though	0	provided that	0
after	7,042	provided	594	both and	0	providing that	0
whether	5,978	whilst	351	but that	0	seeing as	0
for	5,935	suppose	281	but then	0	seeing as how	0
although	5,424	cos	188	but then again	0	seeing that	0
until	5,072	supposing	185	either or	0	without	0

English Word Classes

- Auxillary Verbs:
  - Semantic features of the verb whether the action is present, past or future
  - For eg. *be, do, have, had, are, an, could, may, might, must, shall, should, will, would*

English Word Classes

- Others:
  - interjection: *oh, as, hey, alas, um*
  - negatives: *no, not*
  - politeness markers: *please, thank you*

Part-of-Speech Tagging

- Process of assigning a part of speech pr other syntactic class marker to each word in the corpus
- Tokenisation is required, in general, before PoS Tagging
- Input: A string of words and a special tagset
- Output: A single best tag for each word

Penn Treebank POS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	“to”	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential ‘there’	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*’s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one’s*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

PoS Tagging

- Example of output:
  - *Book/VB that/DT flight/NN*
    - *Book* is ambiguous — it may be NN or VB
  - *Does/VBZ that/DT flight/NN serve/VB dinner/ NN ?/.*
    - *that can be a determiner or complementiser*
      - *Does that flight serve dinner*
      - *I though that it will be rain today*

Part of PoS tagging is to disambiguate

PoS Tagging

- How hard is tagging problem?
  - Good news: Most words in English are unambiguous — that is, they have only one tag
  - Bad news: many of the common English words are ambiguous!
    - can — auxiliary (‘to be able’), a noun or a verb

PoS Tagging

- DeRose (1988) reports that
  - 11.5% words in *Brown corpus* are ambiguous
  - 40% of *Brown tokens* are ambiguous
  - Fortunately many of the 40% ambiguous tokens are easy to disambiguate
  - All choices are not equally likely — *a* will be mostly a determiner than being part of an acronym or an initial

PoS Tagging

- Usage of the word *back*
  - *The back door* = JJ
  - *On my back* = NN
  - *Win the voters back* = RB
  - *Promised to back the bill* = VB

PoS Tagging

		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well*, *beat*)	32	
	7 tags	2	(*still*, *down*)	6	(*well*, *set*, *round*, *open*, *fit*, *down*)
	8 tags			4	(*'s*, *half*, *back*, *a*)
	9 tags			3	(*that*, *more*, *in*)

Rule-based PoS Tagging

- Rule-based tagger — EngCG
  - it is based on the Constraint Grammar architecture of Karlsson et al (1995)

Rule-based PoS Tagging

- The earliest algorithms were based on a two stage architecture (Harris, 1962; Klein and Simmons, 1963; Greene and Rubin, 1971)
  - The first stage used a dictionary to assign each word a list of potential parts-of-speech
  - The second stage used large lists of hand-written disambiguation rules to find the single part-of-speech for each word

Rule-based PoS Tagging

- Modern rule-based approaches to part-of-speech tagging have a similar architecture
- The dictionaries and the rule sets are vastly larger than in the 1960’s
- One of the most comprehensive rule-based approaches is the Constraint Grammar approach (Karlsson et al., 1995a)
- We takeup EngCG tagger for our study (Voutilainen, 1995, 1999)

LNMIIT

EngCG Tagger

- The EngCG ENGTWOL lexicon is based on the two-level morphology and has about 56,000 entries for English word stems
- It counts a word with multiple parts-of-speech (e.g., nominal and verbal senses of hit) as separate entries
- But it does not count the inflected and derived forms
- Each entry is annotated with a set of morphological and syntactic features.

ENGTWOL Lexicon

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

SG for singular,
SG3 for other than third-person-singular.
NOMINATIVE means non-genitive and PCP2 means past participle. PRE, CENTRAL, and POST are ordering slots for determiners (predeterminers (all) come before determiners (the): all the president's men).

NOINDEFDETERMINER means that words like furniture do not appear with the indefinite determiner a. SV, SVO, and SVOO specify the subcategorization or complementation pattern for the verb. SV means the verb appears solely with a subject (nothing occurred); SVO with a subject and an object (I showed the film); SVOO with a subject and two complements: She showed her the ball.

LNMIIT

EngCG Tagger

- In the first stage of the tagger, each word is run through the two-level lexicon transducer
- All the entries for all possible parts-of-speech are returned.
- For example take the phrase:
  - *Pavlov had shown that salivation…*

Pavlov	PAVLOV N NOM SG PROPER
had	HAVE V PAST VFIN SVO
	HAVE PCP2 SVO
shown	SHOW PCP2 SVOO SVO SV
that	ADV
	PRON DEM SG
	DET CENTRAL DEM SG
	CS
salivation	N NOM SG

EngCG Tagger

- EngCG then applies a large set of constraints — 3,744 constraints in the EngCG-2 system for the input sentence
- This constraints rule out incorrect parts-of-speech
- The constraints are used in a negative way, to eliminate tags that are inconsistent with the context
- For example one constraint eliminates all readings of *that* except the ADV (adverbial intensifier) sense (this is the sense in the sentence *it isn't that odd*)

EngCG Tagger

ADVERBIAL-THAT RULE
Given input: "that"
if
(+1 A/ADV/QUANT); /* *if next word is adj, adverb, or quantifier* */
(+2 SENT-LIM); /* *and following which is a sentence boundary,* */
(NOT -1 SVOC/A); /* *and the previous word is not a verb like* */
/* *'consider' which allows adjs as object complements* */
then eliminate non-ADV tags
else eliminate ADV tag

HMM Part-of-Speech Tagging

- Probabilities usage for tagging is quite old:
  - Probabilities in tagging was first used by Stolz et al. (1965)
  - A complete probabilistic tagger with Viterbi decoding was sketched by Bahl and Mercer (1976)
  - Various stochastic taggers were built in the 1980s (Marshall, 1983; Garside, 1987; Church, 1988; DeRose, 1988)
  - We shall study a particular stochastic tagging algorithm generally known as the Hidden Markov Model or HMM tagger

HMM Part-of-Speech Tagging

- Hidden Markov Model is a special case of Bayesian inference
- Bayesian inference or Bayesian classification was applied successfully to language problems as early as the late 1950s, including the OCR work of Bledsoe in 1959, and the seminal work of Mosteller and Wallace (1964) on applying Bayesian inference to determine the authorship of the Federalist papers
- Part-of-speech tagging is a sequence classification task
- The observation is a sequence of words and it is our/system's job to assign them a sequence of part-of-speech tags

HMM Part-of-Speech Tagging

- Let's take this sentence: *Secretariat is expected to race tomorrow*
- Out of all sequences of *n* tags t^n find the one for which $P(t^n|w^n)$ is highest
- $\hat{t}_1^{\,n} = argmax_{t_1} P(t_1^n \mid w_1^n)$
  - *How to find this?*

HMM Part-of-Speech Tagging

$$\hat{t}_1^n = argmax_{t_1^n} P(t_1^n \mid w_1^n)$$

Use the Bayes' rule: $P(x \mid y) = \dfrac{P(y \mid x)P(x)}{P(y)}$

$$\hat{t}_1^n = argmax_{t_1^n} \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = argmax_{t_1^n} \underbrace{P(w_1^n \mid t_1^n)}_{\text{Likelihood}} \underbrace{P(t_1^n)}_{\text{Prior}}$$

HMM Part-of-Speech Tagging

$$\hat{t}_1^{\,n} = argmax_{t_1^n} \, P(w_1^n \mid t_1^n) P(t_1^n)$$

First Assumption:

$P(w_1^n \mid t_1^n) \approx \prod_i^n P(w_i \mid t_i)$ (probability of word depends only on the present tag)

Second Assumption:

$P(t_1^n) \approx \prod_i^n P(t_i \mid t_{i-1})$ (present tag depends only on the previous tag)

HMM Part-of-Speech Tagging

Word Likelihood

tag transition

$$\hat{t}_1^{\,n} = argmax_{t_1^n} P(t_1^n \mid w_1^n) \approx armax_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

LNMIIT

HMM Part-of-Speech Tagging

- Tag transition probabilities:
- Determiners likely to precede adjs and nouns
    - That/DT boy/NN
    - The/DT blue-eyed/JJ boy/NN
    - In general P(NN|DT) and P(JJ|DT) will be high
- Compute P(NN|DT) by counting in a labeled corpus:

LNMIIT

HMM Part-of-Speech Tagging

Compute P(NN|DT) by counting in a labeled corpus:

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

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

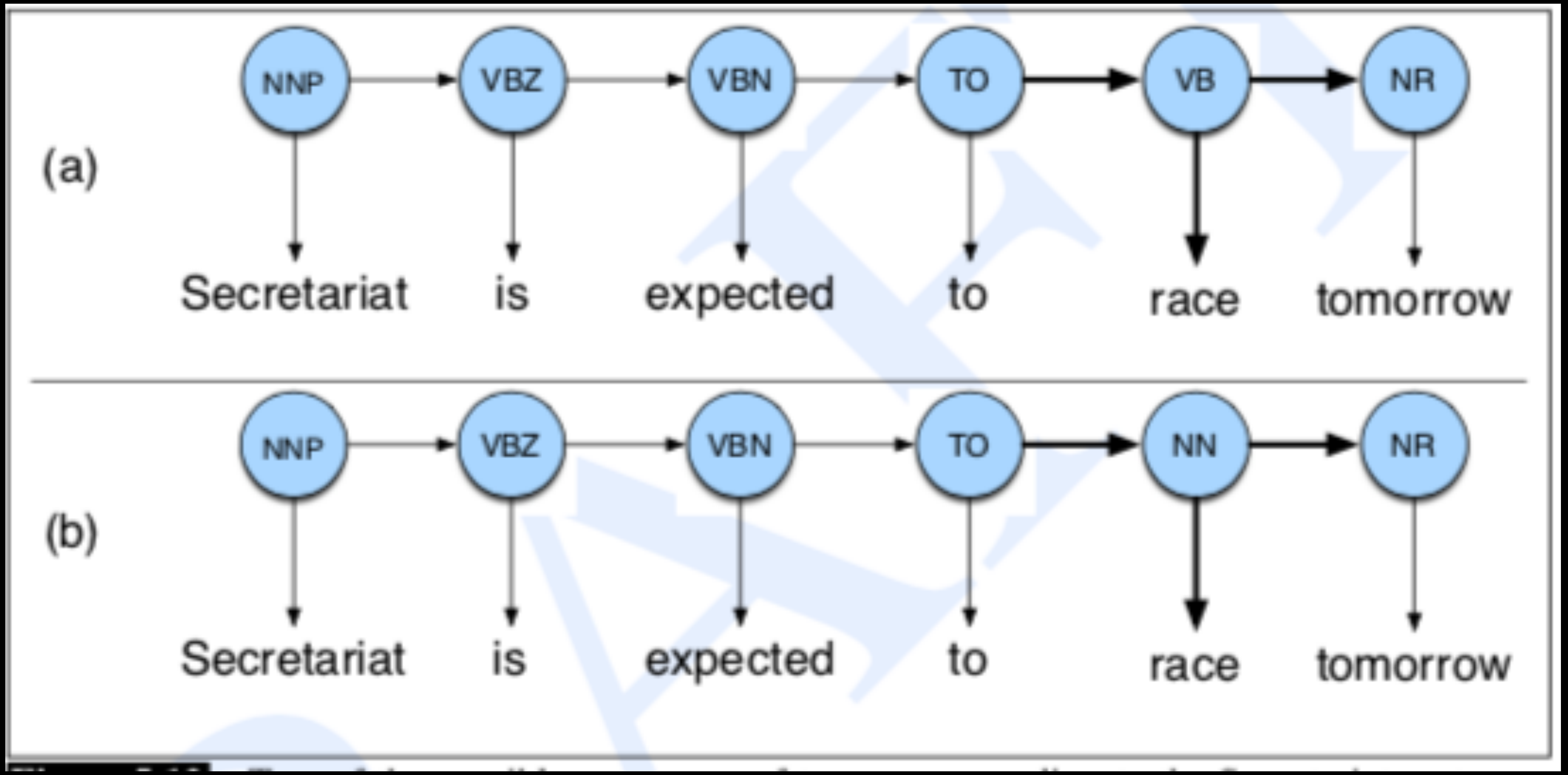
Treebank Brown Corpus

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

LNMIIT

HMM Part-of-Speech Tagging

Secretariat/NNP is/BEZ expected/VBN to/TO **race**/VB tomorrow/NR



Use 87-tag
Brown Corpus
TO for infinitive "to"

$$P(NN \mid TO) = .00047$$

$$P(VB \mid TO) = 0.83$$

$$P(race \mid NN) = 0.00057$$

$$P(race \mid VB) = 0.00012$$

$$P(NR \mid VB) = 0.0027$$

$$P(NR \mid NN) = 0.0012$$

$$P(VB \mid TO)P(NR \mid VB)P(race \mid VB) = 0.00000027$$

$$P(NN \mid TO)P(NR \mid NN)P(race \mid NN) = 0.00000000032$$

Formalising Hidden Markov Model

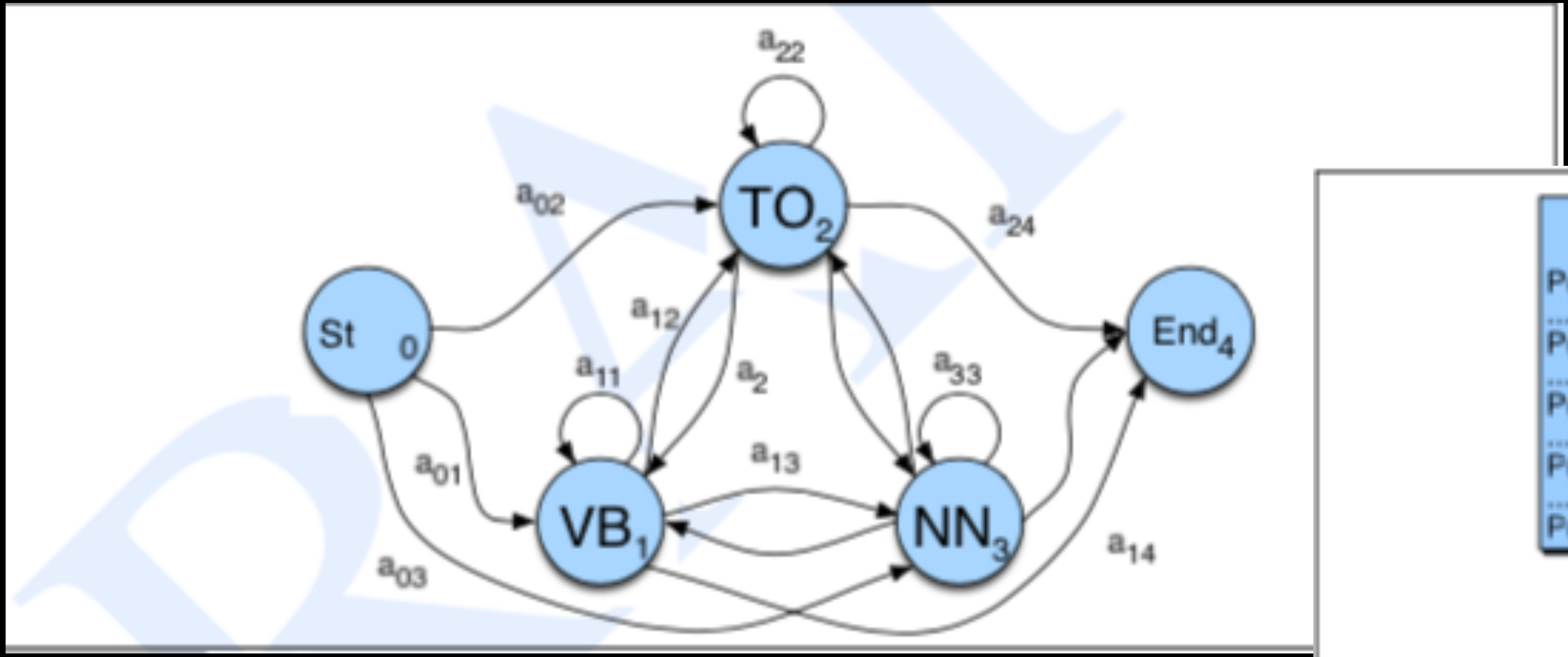
- A weighted finite-state automaton is a simple augmentation of the finite automaton in which each arc is associated with a probability
- The probability on all the arcs leaving a node must sum to 1
- A Markov chain is a special case of a weighted automaton in which the input sequence uniquely determines which states the automaton will go through
- While the Markov chain is appropriate for situations where we can see the actual conditioning events, it is not appropriate in part-of-speech tagging
- This is because in part-of-speech tagging, while we observe the words in the input, we do not observe the part-of-speech tags
- A Hidden Markov Model (HMM) allows us to talk about both observed events (like words that we see in the input) and hidden events (like part-of- speech tags) that we think of as causal factors in our probabilistic model

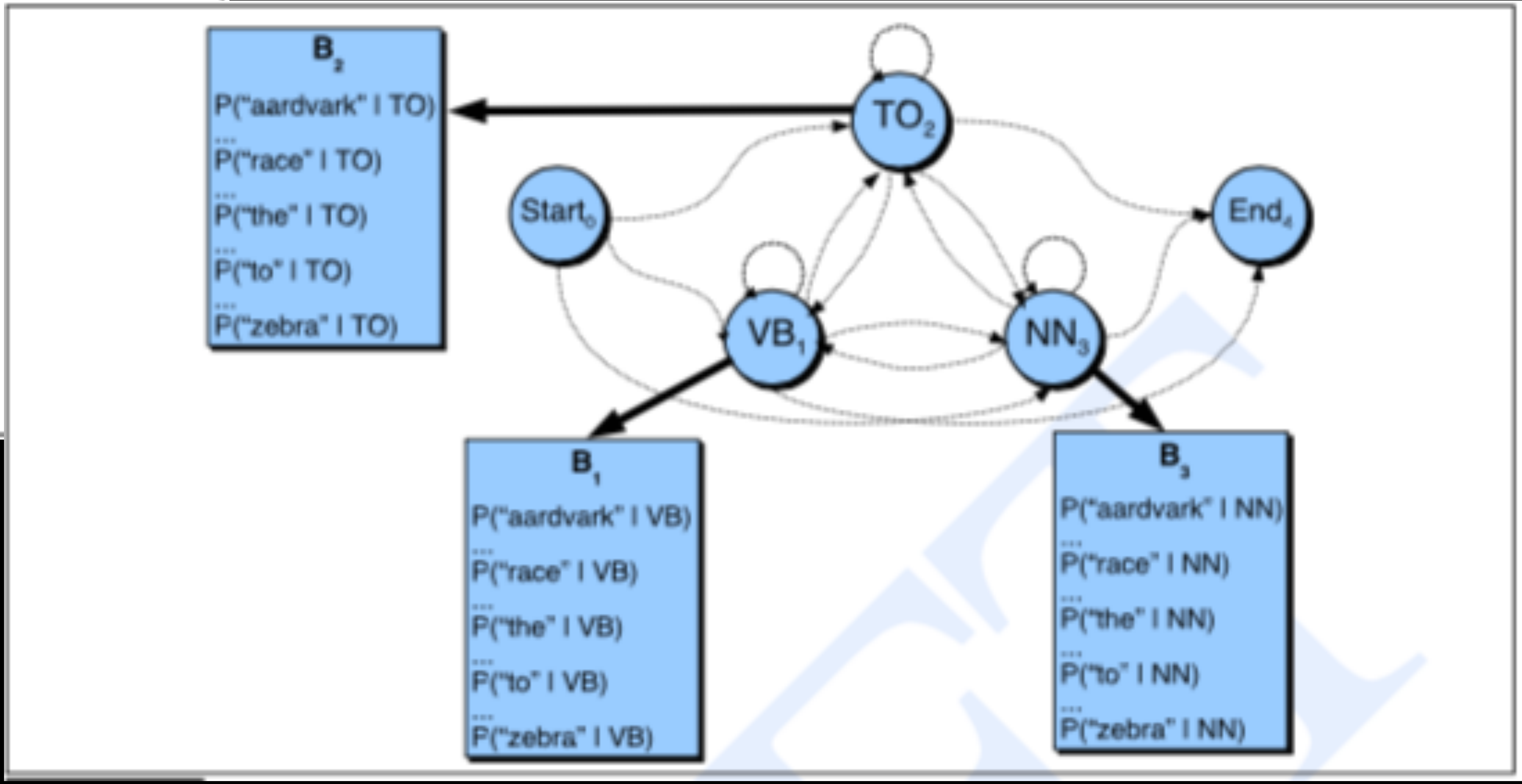
Hidden Markov Model

$Q = q_1 q_2 \ldots q_N$	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** which are not associated with observations, together with transition probabilities $a_{01} a_{02} .. a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state.

LNMIIT

Hidden Markov Model





Hidden Markov Model

- INPUT: $O = (o_1 o_2 \ldots o_T)$ — observed words
- OUTPUT: $Q = (q_1 q_2 \ldots q_T)$ — most probable tag sequence, together with its probability

Hidden Markov Model

- INPUT: $O = (o_1 o_2 \ldots o_T)$

observed words

- OUTPUT: $Q = (q_1 q_2 \ldots q_T)$

most probable tag sequence, together with its probability

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

Transition probabilities between tags (hidden states)

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Observation likelihood from 87-tag brown corpus without smoothing

Hidden Markov Model

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

create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ;initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ;recursion step
for each state s **from** 1 **to** N **do**
$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Hidden Markov Model

- The algorithm first creates N or four state columns
- The first column corresponds to the observation of the first word “I”, the second to the second word “WANT”, the third to the third word “TO”, and the fourth to the fourth word “RACE”
- We begin in the first column by setting the Viterbi value in each cell to the product of the transition probability and the observation probability (of the first word)
- Then we move on, column by column; for every state in column 1, we compute the probability of moving into each state in column 2, and so on.
- For each state q_j at time t, the value viterbi[s,t] is computed by taking the maximum over the extensions of all the paths that lead to the current cell, following the following equation:

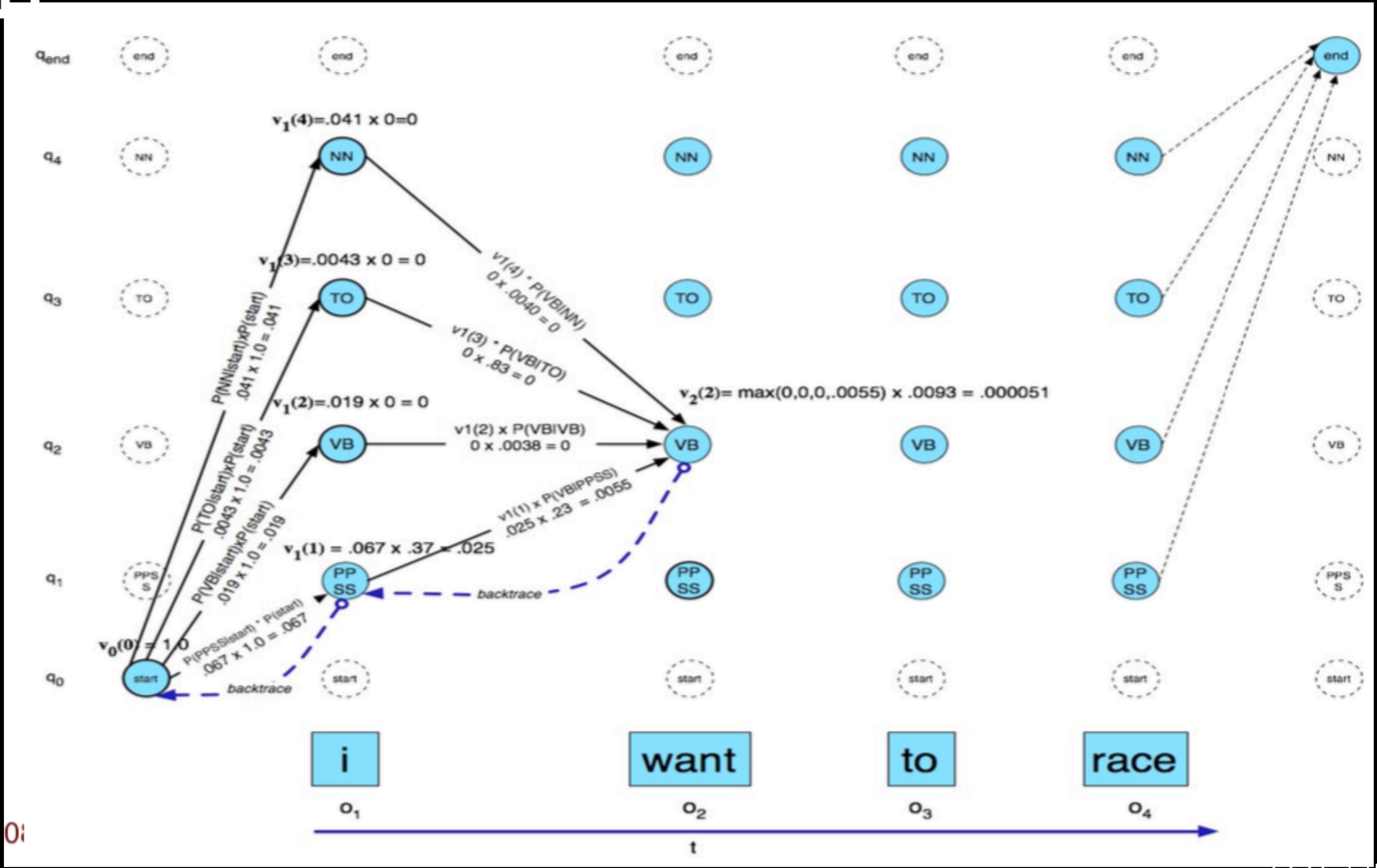
$$v_t(j) = max_{i=1}^{N} v_{t-i}(i) a_{ij} b_j(a_i)$$

previous Viterbi path probability

transition probability

state observation likelihood

Hidden Markov Model



HMM and MEMM

- Hidden Markov Model (HMM)
  - HMMs, forward and Viterbi algorithms
- Maximum Entropy Markov Model (MEMM)
  - For supervised or semi-supervised learning

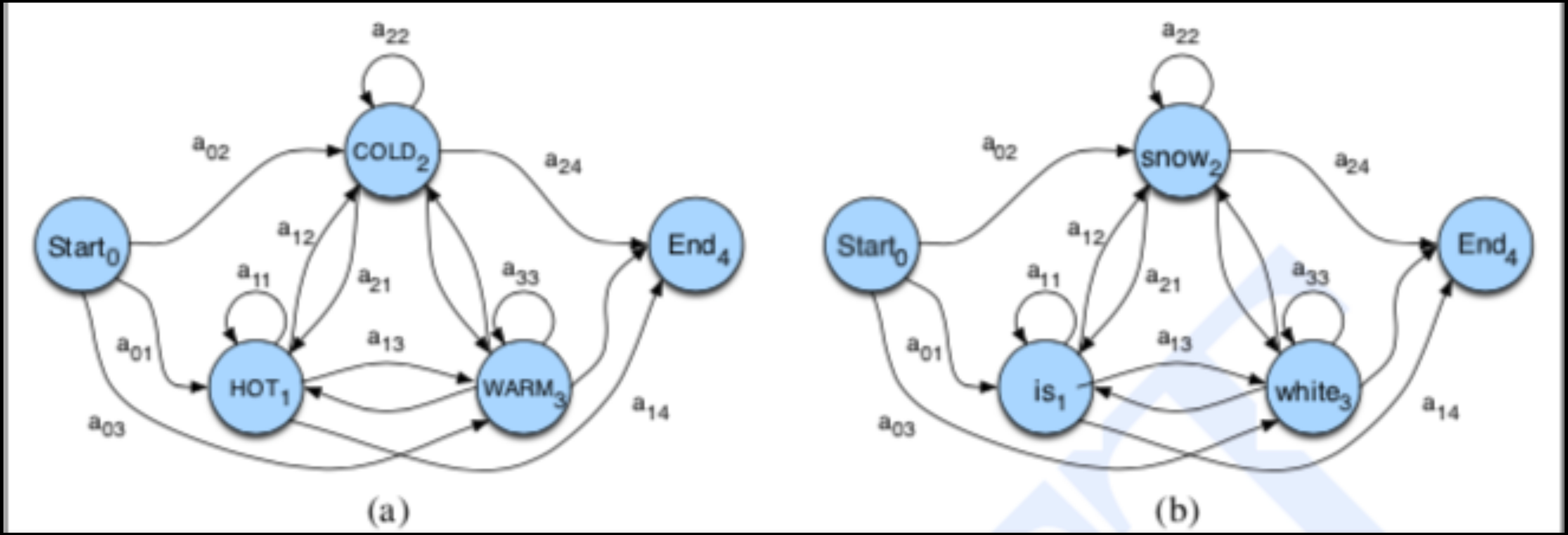
Both are machine learning models.

They are sequence classifiers/sequence labellers

HMM and MEMM

- A weighted finite automaton (WFA) is a finite automaton with weights defined on the transitions
- The weight is nothing but the probability score
- Weight indicated how likely that path will be taken by the automaton
- Probability of all the transitions leaving a state should sum upto 1
- Markov chain is a special case of WFA where the input sequence uniquely determines which states it goes through

Markov Chain



Markov Chain

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
q_0, q_F	a special **start state** and **end (final) state** which are not associated with observations.

- Probabilistic Graphical Model / Automaton model
- Markov Assumption: $P(q_i \mid q_1 q_2 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$
- Note: $\sum_{j=1}^{n} a_{ij} = 1 \forall i$

LNMIIT

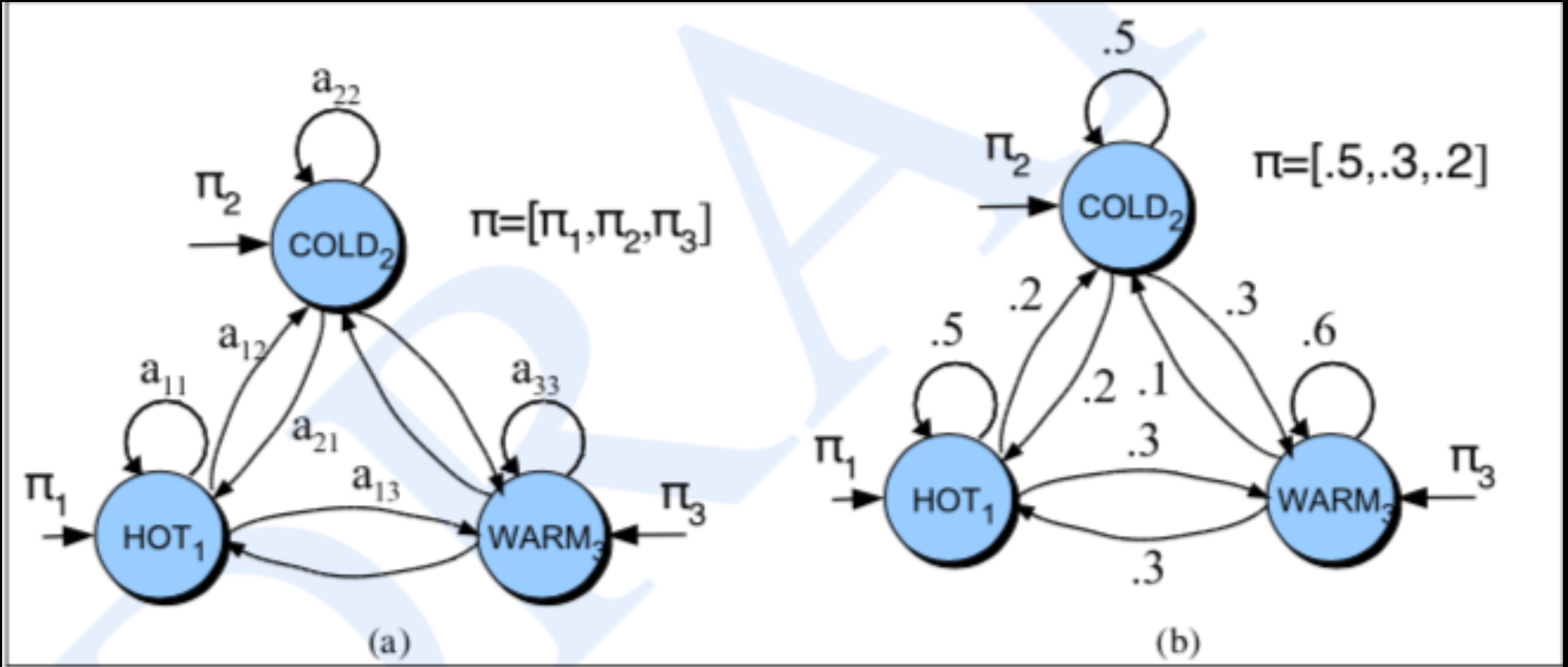
Markov Chain

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
q_0, q_F	a special **start state** and **end (final) state** which are not associated with observations.

$\pi = \pi_1, \pi_2, \ldots, \pi_N$	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, $\sum_{i=1}^{n} \pi_i = 1$
$QA = \{q_x, q_y \ldots\}$	a set $QA \subset Q$ of legal **accepting states**

where $\sum_{i=1}^{n} \pi_i = 1$

Markov Chain



(1) Compute the probability of *hot hot hot hot*

(2) *cold hot cold hot*

Hidden Markov Model

- Markov chain is good for finding the probability of an event or a chain of event that we can observe in the world
- But in reality events will not be directly observable!
- PoS Tags are not directly observable
- What we can see are the sequence of words not tags!
- PoS tags are hidden behind words — that is why we can this as Hidden Markov Model
- HMM gives flexibility to talk about both the observed events and the the hidden events

Hidden Markov Model

$Q = q_1 q_2 \ldots q_N$	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** which are not associated with observations, together with transition probabilities $a_{01} a_{02} .. a_{0n}$ out of the start state and $a_{1F} a_{2F} ... a_{nF}$ into the end state.

Hidden Markov Model

$Q = q_1 q_2 \ldots q_N$	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** which are not associated with observations, together with transition probabilities $a_{01} a_{02} .. a_{0n}$ 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**

Hidden Markov Model

Markov Assumption:

$$P(q_i \mid q_1 q_2 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

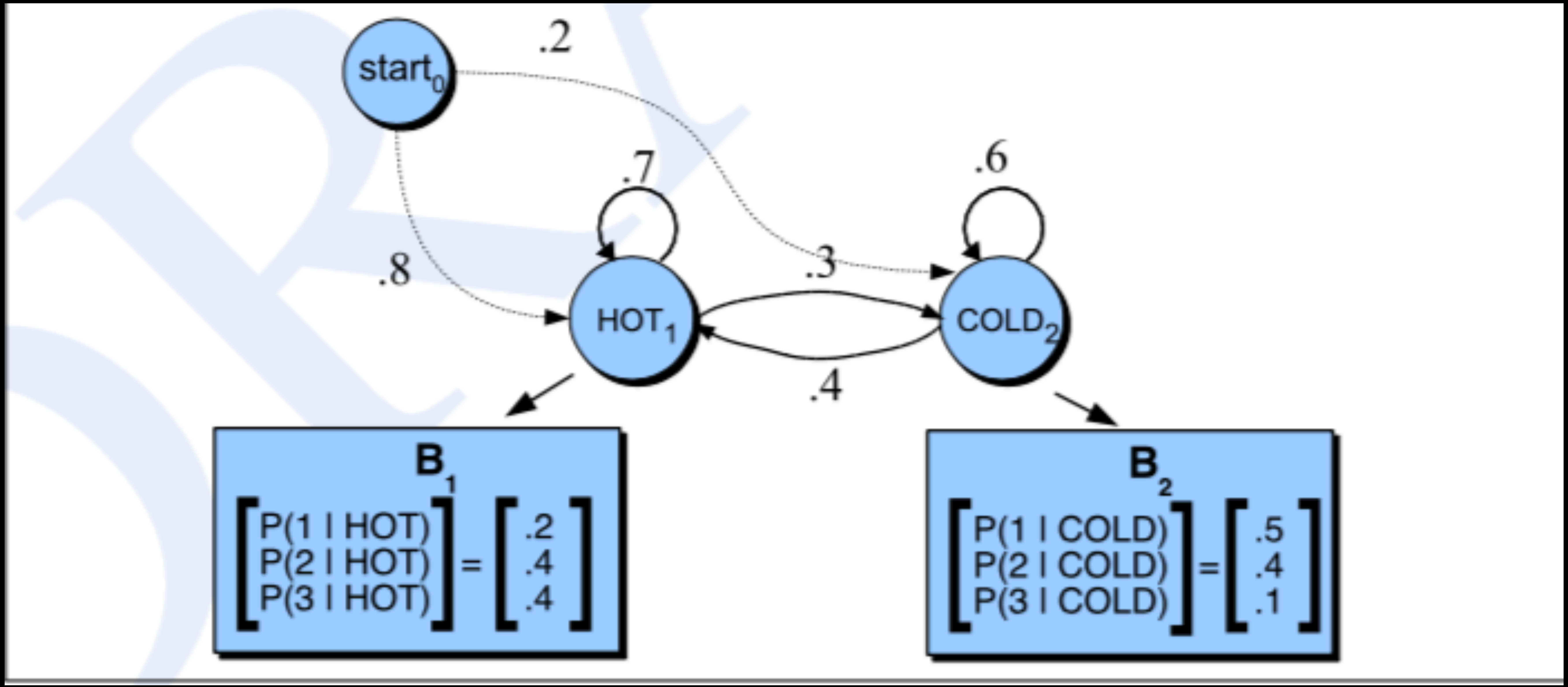
Output Independence

$$P(o_i \mid q_1 \ldots q_i \ldots q_T, o_1, \ldots, o_i \ldots, o_T) = P(o_i \mid q_i)$$

Hidden Markov Model

- Task conceived of by Jason Eisner (2002a)
- You are a climatologist in the year 2799 studying the history of global warming
- You cannot find any records of the weather in Baltimore, Maryland, for the summer of 2007, but you do find Jason Eisner's diary, which lists how many ice creams Jason ate every day that summer
- Our goal is to use these observations to estimate the temperature every day. We'll simplify this weather task by assuming there are only two kinds of days: cold (C) and hot (H)
- Given a sequence of observations O, each observation an integer corresponding to the number of ice creams eaten on a given day, figure out the correct 'hidden' sequence Q of weather states (HOT or COLD) which caused Jason to eat the ice cream

Hidden Markov Model



Fully connected or ergodic HMM

Another model is called Bakis — where it is not fully connected

Hidden Markov Model

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