Natural Language Processing

3: Part-of-Speech Tagging

Quiz

pen and paper

Q1: Give 3-5 examples of applications of the Edit Distance Algorithm

Objectives

to be able to answer questions

(1) What are word classes and why are they useful?

(2) What Tagsets for English exist?

(3) What techniques are used in NLP?

(4) Why is Part-of-Speech Tagging Difficult?

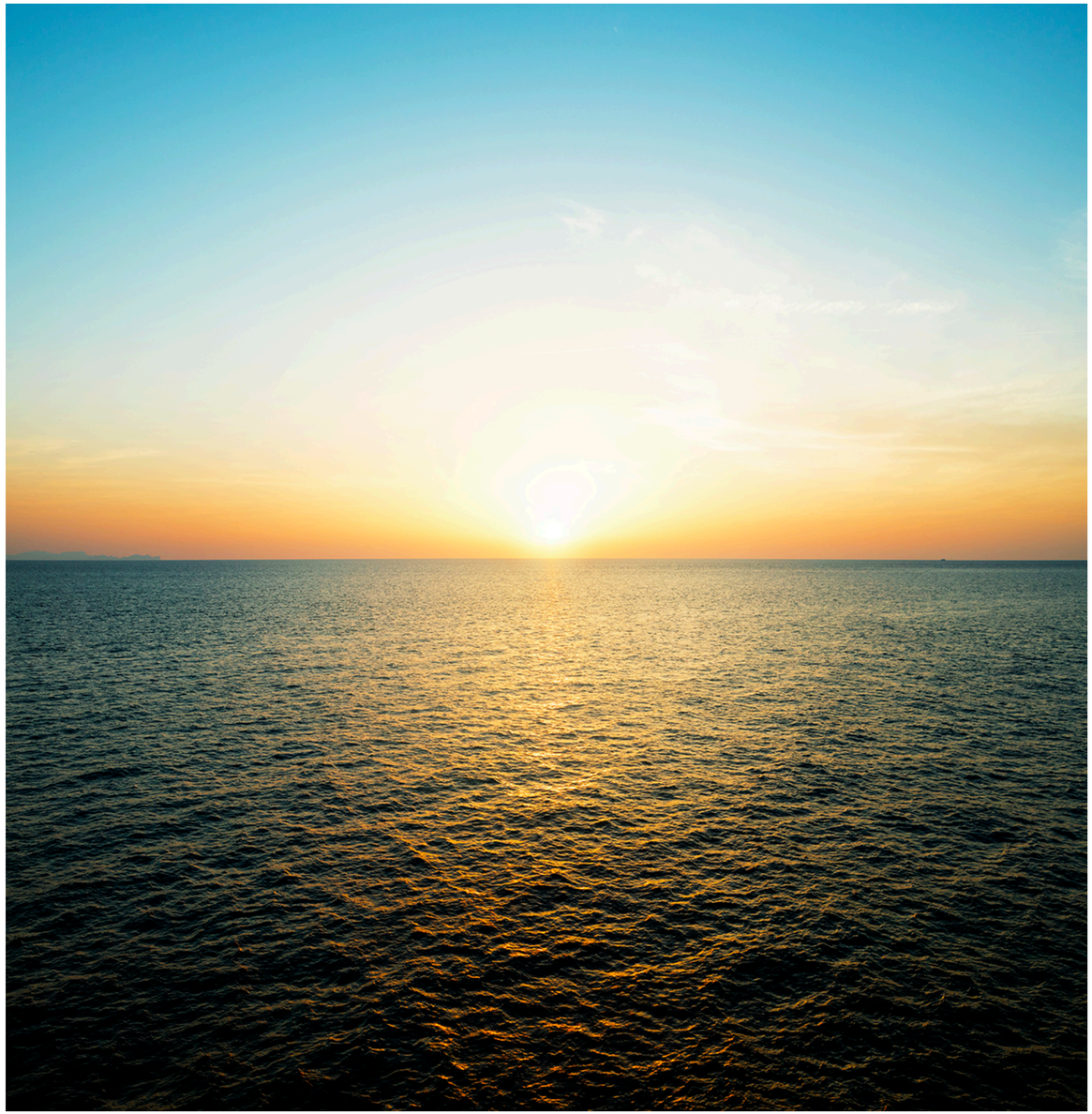
(5) Learn Methods for Part-of-Speech Tagging

   (1) Rule-based Tagging

   (2) Hidden Markov Model-based Tagging

   (3) Conditional Random Fields

Word Classes



Parts of Speech

The idea that words can be classified into grammatical categories

- part of speech, word classes, POS, POS tags

8 parts of speech attributed to Dionysius Thrax of Alexandria (c. 1st C. BCE):

- noun, verb, pronoun, preposition, adverb, conjunction, participle, article
- These categories are relevant for NLP today.

https://web.stanford.edu/~jurafsky/slp3/

Two classes of words: Open vs. Closed

Closed class words

- Relatively fixed membership
- Usually **function** words: short, frequent words with grammatical function
  - determiners: ***a, an, the***
  - pronouns: ***she, he, I***
  - prepositions: ***on, under, over, near, by, …***

Open class words

- Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
  - Plus interjections: **oh, ouch, uh-huh, yes, hello**
- New nouns and verbs like *iPhone* or *to fax*

Word Classes

Definition

(1) Word classes (parts-of-speech, POS) may be defined in terms of

  (1) syntactic and morphological distributional properties
    • words that occur in the same constructions or take the same affixes are in the same class

  (2) semantic function
    • words that perform the same semantic function are in the same class

(2) Syntactic/morphological distributional regularities tend to be favoured over semantic regularities as a basis for defining word classes.

Word Classes

Semantic Function

(1) Words fall into the same class if they perform the same semantic function.

  (1) Nouns perform the semantic function of _____

  (2) Adjectives perform the semantic function of _____

(2) "The black computer is new" computer identifies an object and black qualifies it.

(3) But, compare:

- *Computer science is dull.*
- *Jet black is her favourite colour.*

Word Classes

Distributional regularities

(1) Words fall into the same class if they can appear in the same constructions.

(3) Consider the construction: *It was very* ___

- *It was very new.*
- *It was very black.*
- * *It was very computer.*

Word Classes

Applications 1

(1) A word's class may determine its pronunciation

(2) A word's class suggests which morphological affixes it can take

(3) Certain word classes more useful than others in suggesting a document's content

(4) Shallow parsing can be accomplished using patterns of POS tags.

  (1) **[ADJ** + **NOUN** + **NOUN]** can be used for terminology extraction

  (2) [**Mr. PNOUN PNOUN]** can be used for person name recognisers

Word Classes

Applications 2

(1) Word sense disambiguation assisted by knowing POS

(2) Word class of current word helps predict word class of next

(3) Parsers typically require word class information in order to discover constituency structure.

(4) POS-tagged corpora useful in linguistic research

Open class ("content") words

Nouns

Proper

Janet
Italy

Common

cat, cats
mango

Verbs

Main

eat
went

Auxiliary

can
had

Adjectives *old green tasty*

Adverbs *slowly yesterday*

Numbers

122,312
one

Interjections *Ow hello*

… more

Closed class ("function")

Determiners *the some*

Conjunctions *and or*

Pronouns *they its*

Prepositions *to with*

Particles *off up*

… more

https://web.stanford.edu/~jurafsky/slp3/

Part-of-Speech Tagging

Assigning a part-of-speech to each word in a text.

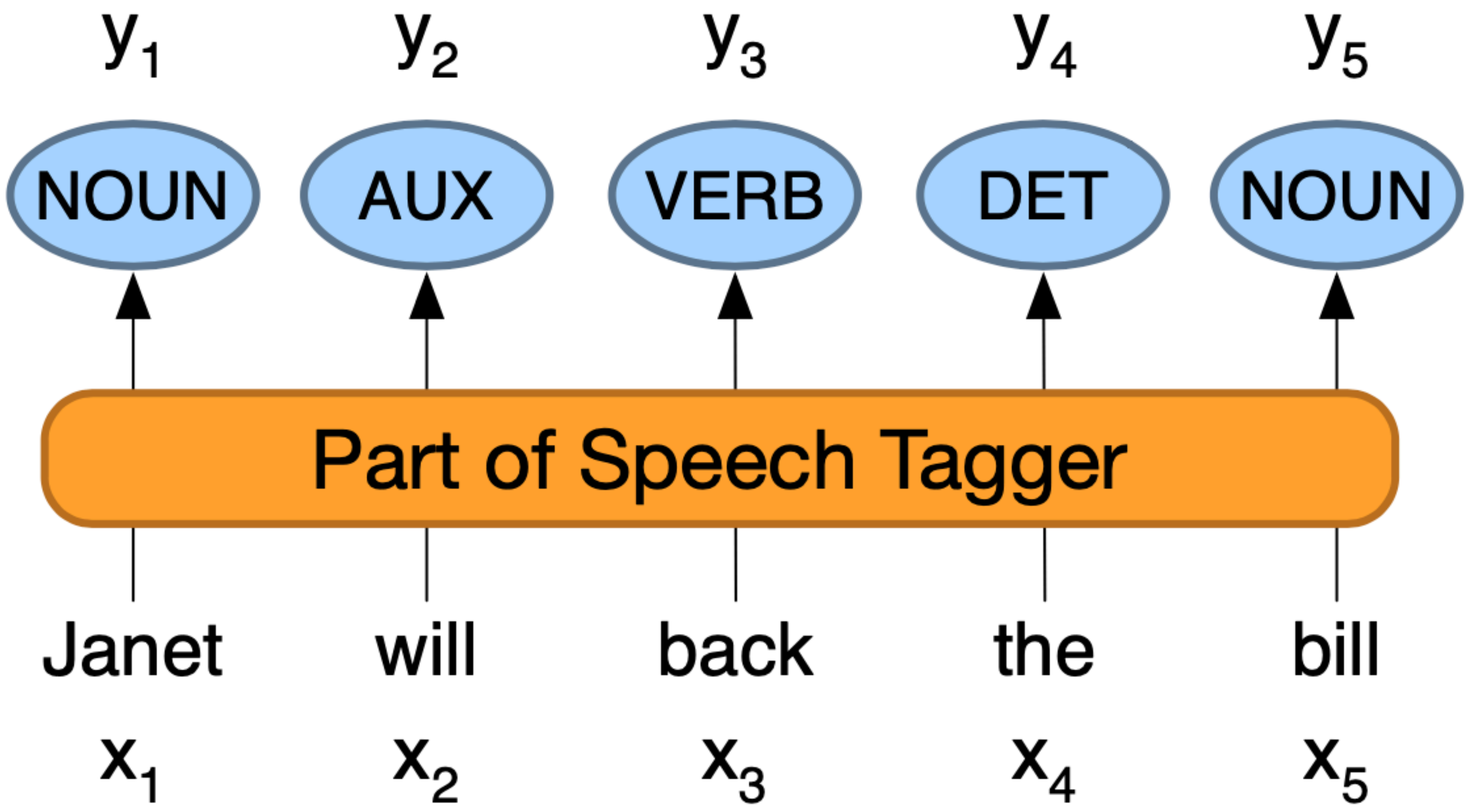
Words often have more than one POS.

book:

- VERB: (***Book*** *that flight*)
- NOUN: (*Hand me that* ***book***).

Part-of-Speech Tagging

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



https://web.stanford.edu/~jurafsky/slp3/

How difficult is POS tagging in English?

Roughly 15% of word types are ambiguous

- Hence 85% of word types are unambiguous
- *Janet* is always PROPN, *hesitantly* is always ADV

But those 15% tend to be very common.

So ~60% of word tokens are ambiguous

E.g., *back*

earnings growth took a back/ADJ seat
a small building in the back/NOUN
a clear majority of senators back/VERB the bill
enable the country to buy back/PART debt
I was twenty-one back/ADV then

https://web.stanford.edu/~jurafsky/slp3/

POS tagging performance in English

How many tags are correct? (Tag accuracy)

- About 97%
- Hasn't changed in the last 10+ years
- HMMs, CRFs, BERT perform similarly .
- Human accuracy about the same

But baseline is 92%!

- Baseline is performance of stupidest possible method
- "Most frequent class baseline" is an important baseline for many tasks
  - Tag every word with its most frequent tag
  - (and tag unknown words as nouns)
- Partly easy because
- Many words are unambiguous

Sources of information for POS tagging

```
Janet will back the bill
  AUX/NOUN/VERB?     NOUN/VERB?
```

Prior probabilities of word/tag

- "will" is usually an AUX

Identity of neighboring words

- "the" means the next word is probably not a verb

Morphology and wordshape:

- Prefixes — unable: — un- → ADJ
- Suffixes — importantly: — -ly → ADJ
- Capitalization — Janet: — CAP → PROPN

Standard algorithms for POS tagging

Supervised Machine Learning Algorithms:

- Hidden Markov Models
- Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)
- Large Language Models (like BERT), fine-tuned

All required a hand-labeled training set, all about equal performance (97% on English)

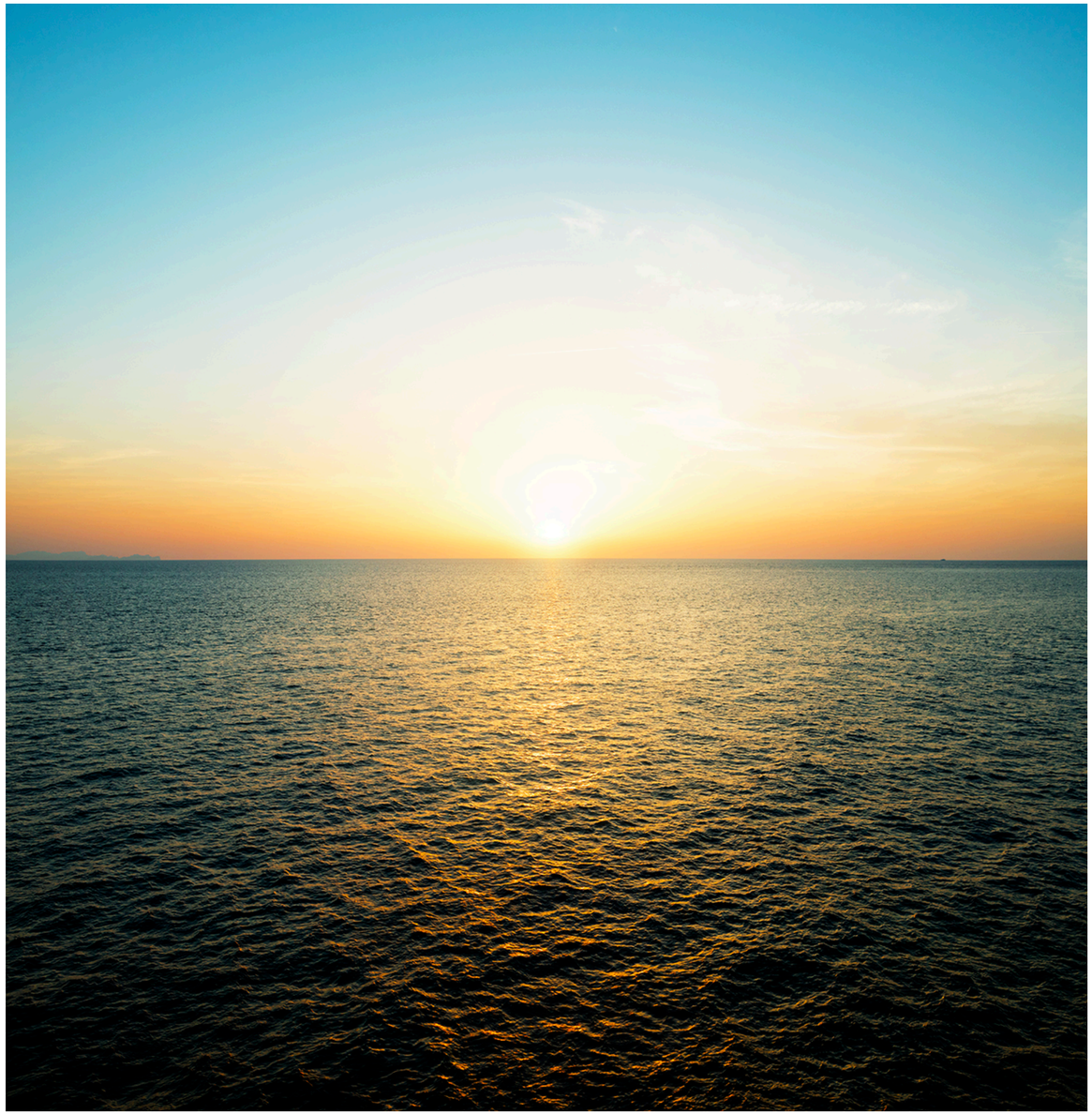
All make use of information sources we discussed

- Via human created features: HMMs and CRFs
- Via representation learning: Neural LMs

https://web.stanford.edu/~jurafsky/slp3/

Tagsets for English

(1) and for Russian



Tagsets for English

Part-of-speech (POS) tagging is critically dependent on the inventory of POS tags

(1) A number of POS tagsets have been developed:

  (1) **Brown tagset 87-tag** tagset developed for use with the Brown Corpus

  (2) **Penn Treebank tagset 45-tag** tagset developed for use with the Penn Treebank

  (3) **C5 tagset 61-tag** tagset used to tag the British National Corpus

  (4) **C7 tagset 146-tag** tagset also used to tag the British National Corpus

Example Tagged Text

An example sentence tagged with Penn Treebank tags (from the Treebank)

The/DT trade/NN gap/NN is/VBZ expected/VBN to/TO widen/VB to/TO about/IN $/$ 9/CD billion/CD from/IN July /NNP 's/POS $/$ 7.6/CD billion/CD ,/, according/VBG to/TO a /DT survey/NN by IN MMS/NNP International/NNP ,/, a /DT unit/NN of/IN McGraw-Hill/NNP Inc./NNP ,/, New/NNP York/NNP ./.

Example

(1) The Penn Treebank part-of-speech tags, including punctuation.

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	: ; ... – -
PR	Particle	up, off			

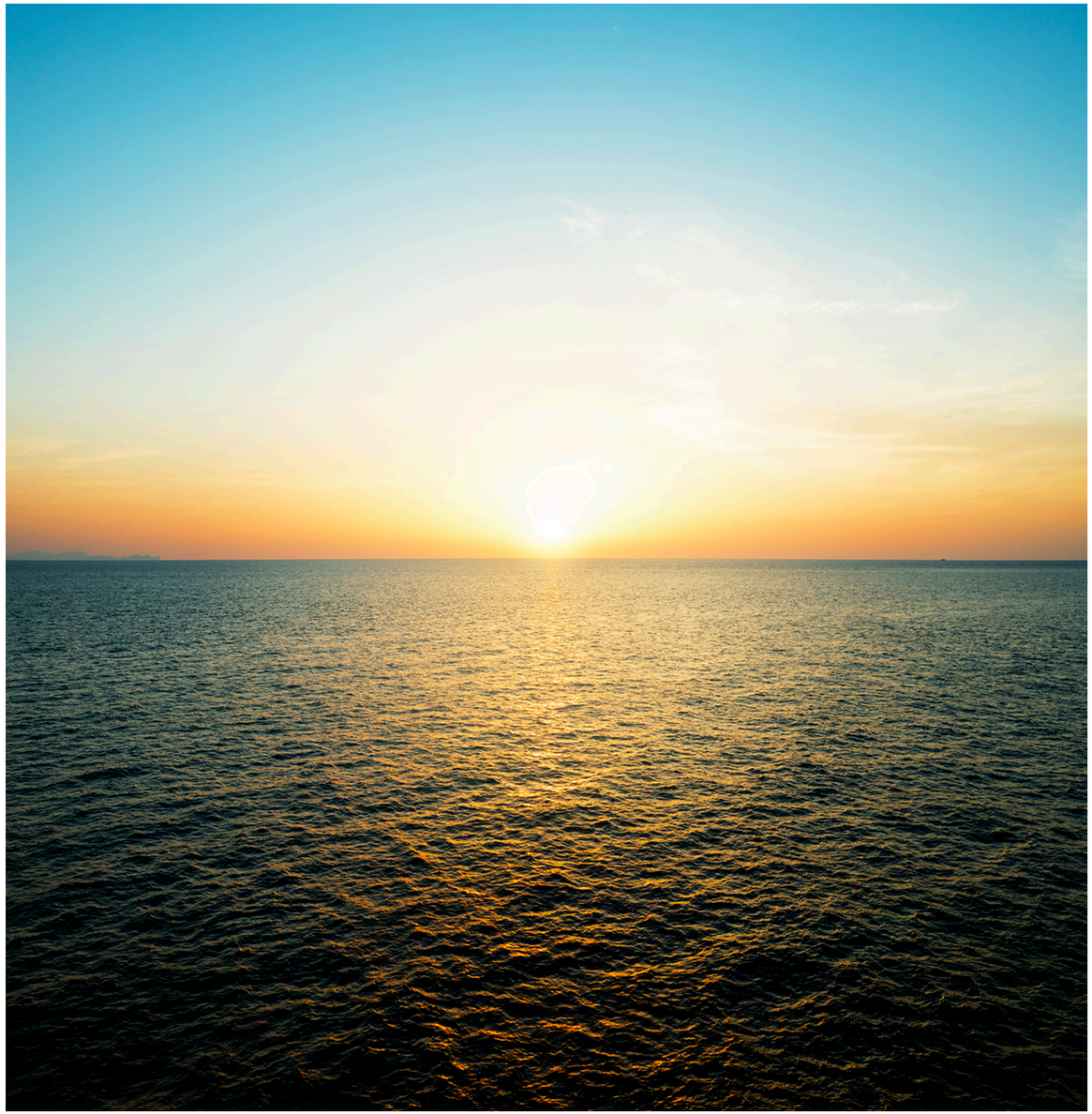
Russian POS tags

Based on ruscorpora.ru: https://ruscorpora.ru/new/corpora-morph.html

Части речи

S — существительное (*яблоня, лошадь, корпус, вечность*)
A — прилагательное (*коричневый, таинственный, морской*)
NUM — числительное (*четыре, десять, много*)
ANUM — числительное-прилагательное (*один, седьмой, восьмидесятый*)
V — глагол (*пользоваться, обрабатывать*)
ADV — наречие (*сгоряча, очень*)
PRAEDIC — предикатив (*жаль, хорошо, пора*)
PARENTH — вводное слово (*кстати, по-моему*)
SPRO — местоимение-существительное (*она, что*)
APRO — местоимение-прилагательное (*который, твой*)
ADVPRO — местоименное наречие (*где, вот*)
PRAEDICPRO — местоимение-предикатив (*некого, нечего*)
PR — предлог (*под, напротив*)
CONJ — союз (*и, чтобы*)
PART — частица (*бы, же, пусть*)
INTJ — междометие (*увы, батюшки*)

Why is POS Tagging Difficult?



How difficult is the problem?

Most words in English have only one tag; but many of the most common words have multiple tags.

(1) For example in the Brown corpus:

  (1) 11.5% of word types are ambiguous

  (2) >40% of word tokens are ambiguous

	Original 87-tag corpus		Treebank 45-tag corpus	
1 tag	44,019		38,857	
2 tags	4,967		8844	
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*)

Types:		**WSJ**	**Brown**
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

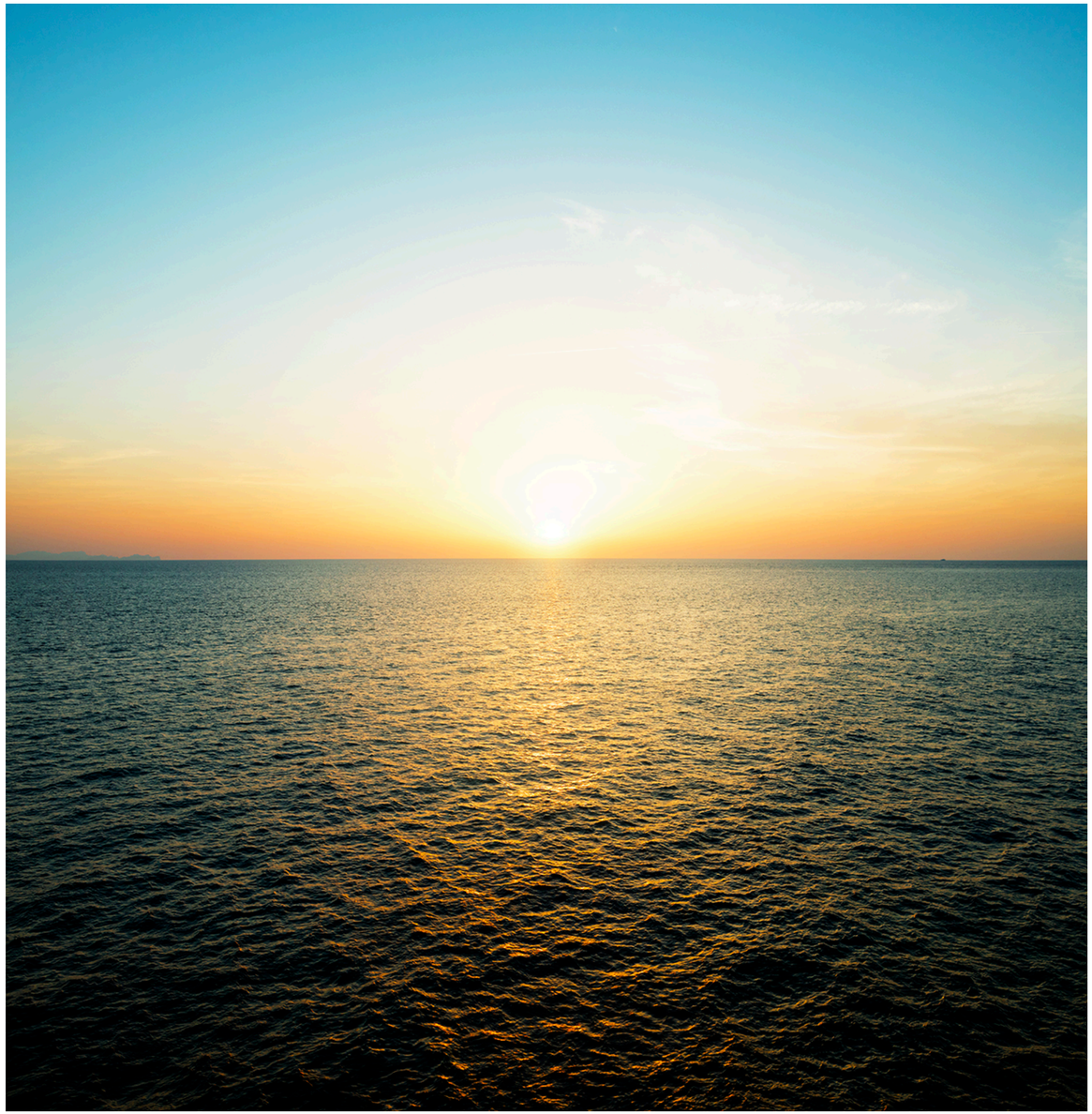
Break

Methods for POS Tagging

(1) rule-based

(2) Hidden Markov Model (HMM)

(3) CRF



Problem setup

(1) **Input:**

A sequence of words that is

(1) tokenised (e.g. punctuation separated from words)

(2) split into sentences (most taggers tag one sentence at a time)

(2) Output:

(1) is the same sequence of words with a single best tag added to each word.

Example

Problem setup

(1) The trade gap is expected to widen to about $ 9 billion from July 's $ 7.6 billion , according to a survey by MMS International , a unit of McGraw-Hill Inc. , New York .

(3) The/DT trade/NN gap/NN is/VBZ expected/VBN to/TO widen/VB to/TO about/IN $/$ 9/CD billion/CD from/IN July/NNP 's/POS $/$ 7.6/CD billion/CD ,/, according/VBG to/TO a/DT survey/NN by/IN MMS/NNP International/NNP ,/, a/DT unit/NN of/IN McGraw-Hill/NNP Inc./NNP ,/, New/NNP York/NNP ./.

HMM

(1) Probabilistic tagging dates back to 1965

(2) Hidden Markov Models (HMMs) are a widely explored approach

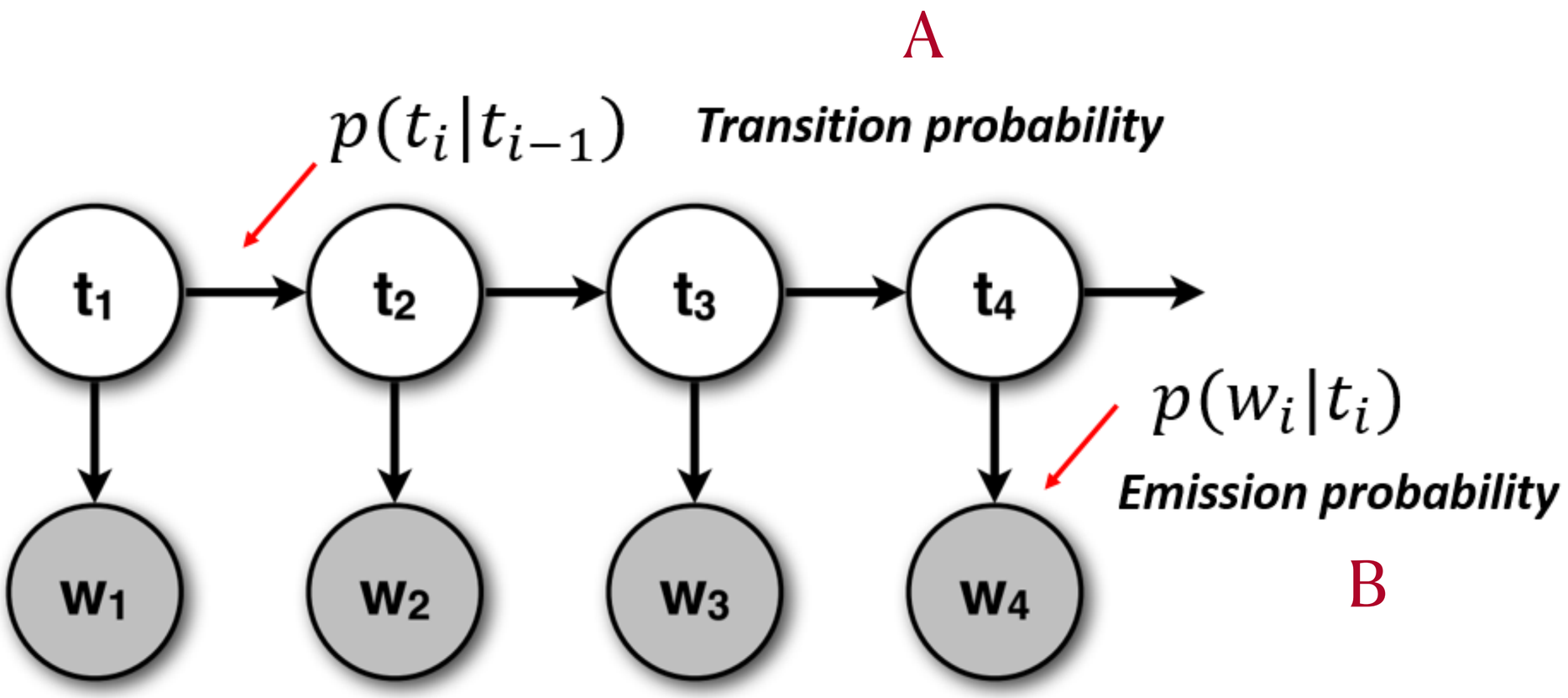
(3) View use of HMMs for POS tagging as a special case of Bayesian inference

(4) POS tagging can be treated as **a sequence labeling** task:
- Observe a sequence of words and assign them a sequence of POS tags

HMM definition

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	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 $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	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, $\sum_{i=1}^{n} \pi_i = 1$

HMM


A
$p(t_i|t_{i-1})$
Transition probability
t1
t2
t3
t4
$p(w_i|t_i)$
Emission probability
B
w1
w2
w3
w4

HMM

(1) What is the best sequence of tags corresponding to a sequence of words?

(2) Of all possible tag sequences t_1^n to assign to the word sequence w_1^n we want the one that is most probable given the observed word sequence w_1^n i.e. we want the tag sequence such that $\mathrm{P}(t_1^n|w_1^n)$ is highest.

(3) Using $\hat{t}_1^n$ denote our estimate of the best tag sequence we have:

$\hat{t}_1^n = \arg\max \mathrm{P}(t_1^n|w_1^n)$

(4) How can we compute $\mathrm{P}(t_1^n|w_1^n)$ for a given tag sequence t_1^n and word sequence w_1^n?

HMM

- Use Bayes' rule: **P(T|W) = P(W|T)P(T) / P(W)**

- to transform this into something which is easier to compute:

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

  = **argmax P(**w_1^n**|**t_1^n**)P(**t_1^n**)**

- In words: the most probable tag sequence $\hat{t}_1^n$ given a word sequence w_1^n can be computed by taking the product of two probabilities for each tag sequence and choosing the tag sequence for which this probability is greatest.

HMM

(1) In the equation

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

(2) The two terms are:

The **prior probability** of the tag sequence $P(t_1^n)$; and

The **likelihood** of the word string $P(w_1^n|t_1^n)$

(4) But this argmax $P(w_1^n|t_1^n)P(t_1^n)$ is still hard to compute. (**Why?**)

Complexity of likelihood estimation in HMM

(1) O - output sequence, Q - tag sequence

$$P(O) = \sum_{Q} P(O, Q) = \sum_{Q} P(O \mid Q) P(Q)$$

complexity is approximately $O(N^T)$

N=100, T=25 (=> 10^{50})

HMM

HMM taggers make two simplifying assumptions

(1) the probability of a word appearing is dependent only on its own POS tag and is independent of other words and tags around it. i.e.

$$\mathrm{P}(w_1^n|t_1^n) \approx \Pi_{i=1}^n P(w_i \mid t_i)$$

(2) The probability of a tag appearing is dependent only on the preceding tag (the **bigram** assumption, from the **markovian** property of the model).

$$\mathrm{P}(t_1^n) \approx \Pi_{i=1}^n P(t_i \mid t_{i-1})$$

(1) $$\hat{t}_1^n = \arg\max \mathrm{P}(t_1^n|w_1^n) = \operatorname{argmax} \mathrm{P}(w_1^n|t_1^n)\mathrm{P}(t_1^n) \approx \operatorname{argmax} \Pi_{i=1}^n [P(w_i \mid t_i)P(t_i \mid t_{i-1})]$$

Computing the Most Likely Tag Sequence: Example

(1) $P(w_i \mid t_i) = \dfrac{f(w_i, t_i)}{f(t_i)}$

$f(w, t)$ is the number of occurrences of word w with tag t

(2) $P(t_i \mid t_{i-1} \ldots t_{i-k}) = \dfrac{f(t_{i-k} \ldots t_i)}{f(t_{i-k} \ldots t_{i-1})}$

$f(t_{l_1} \ldots t_{l_m})$ is the number of occurrences of the tag sequence $t_{l_1} \ldots t_{l_m}$

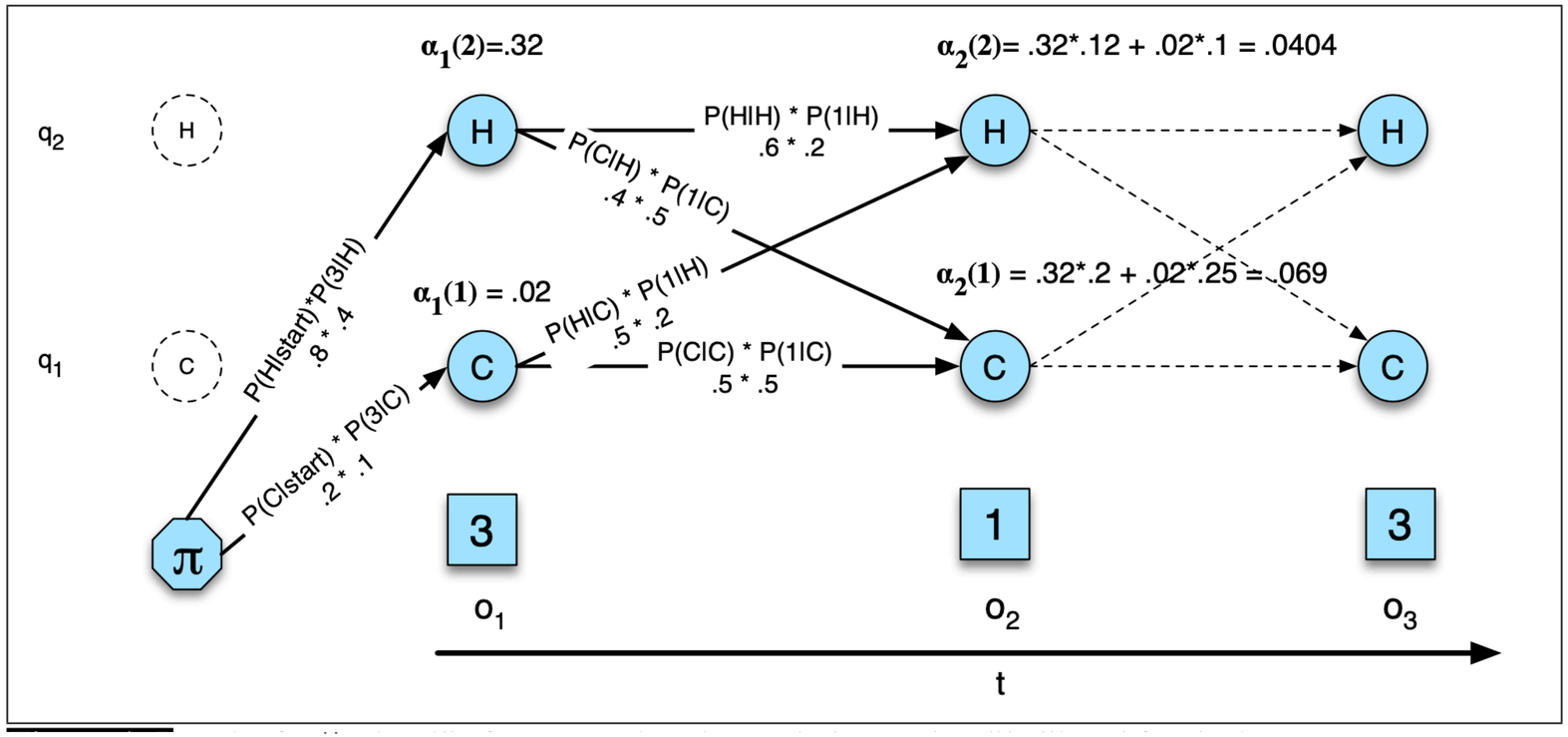
HMMs with Examples

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.

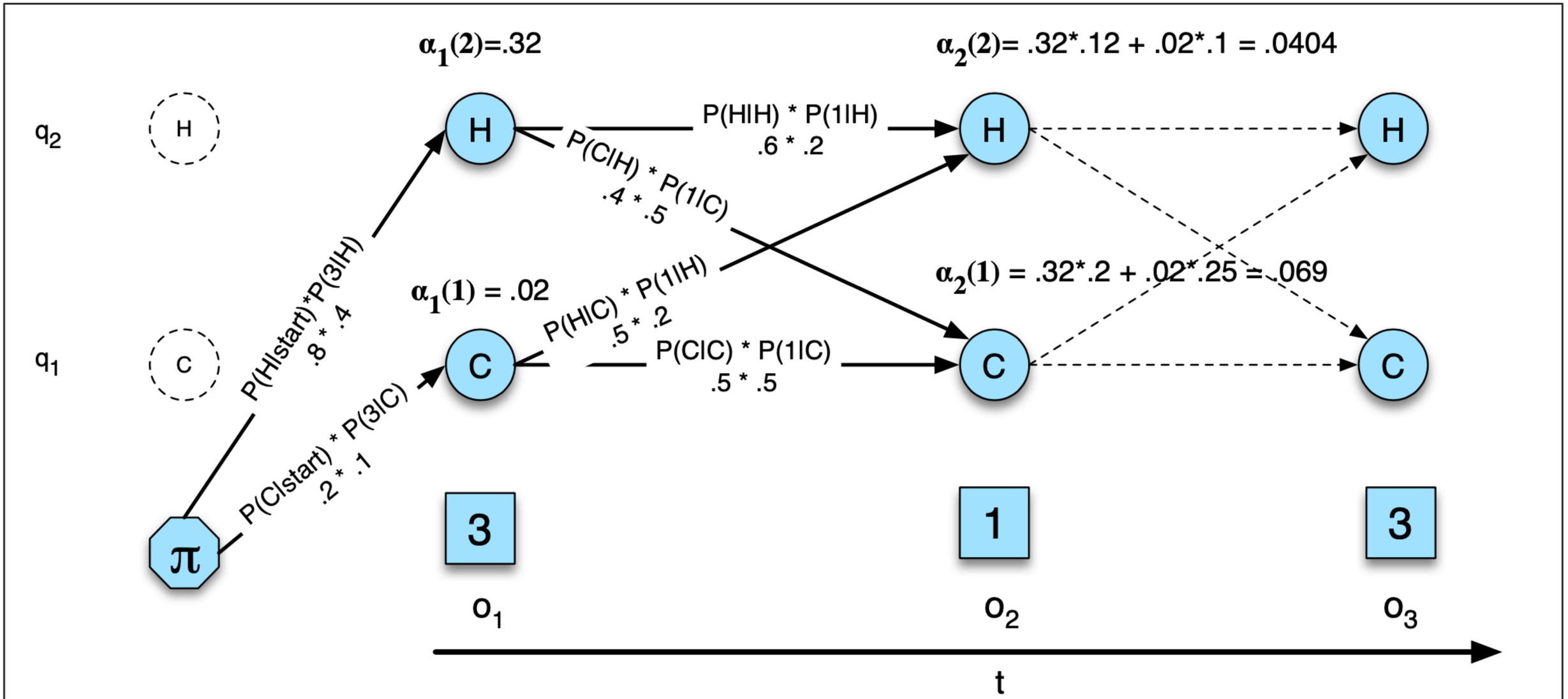
HMM Forward algorithm

(1) $\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j \mid \lambda)$

(2) $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$



HMM Forward algorithm



HMM Forward algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$; termination step

return *forwardprob*

1. Initialization:

$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \le j \le N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

HMM Decoding

Viterbi Algorithm

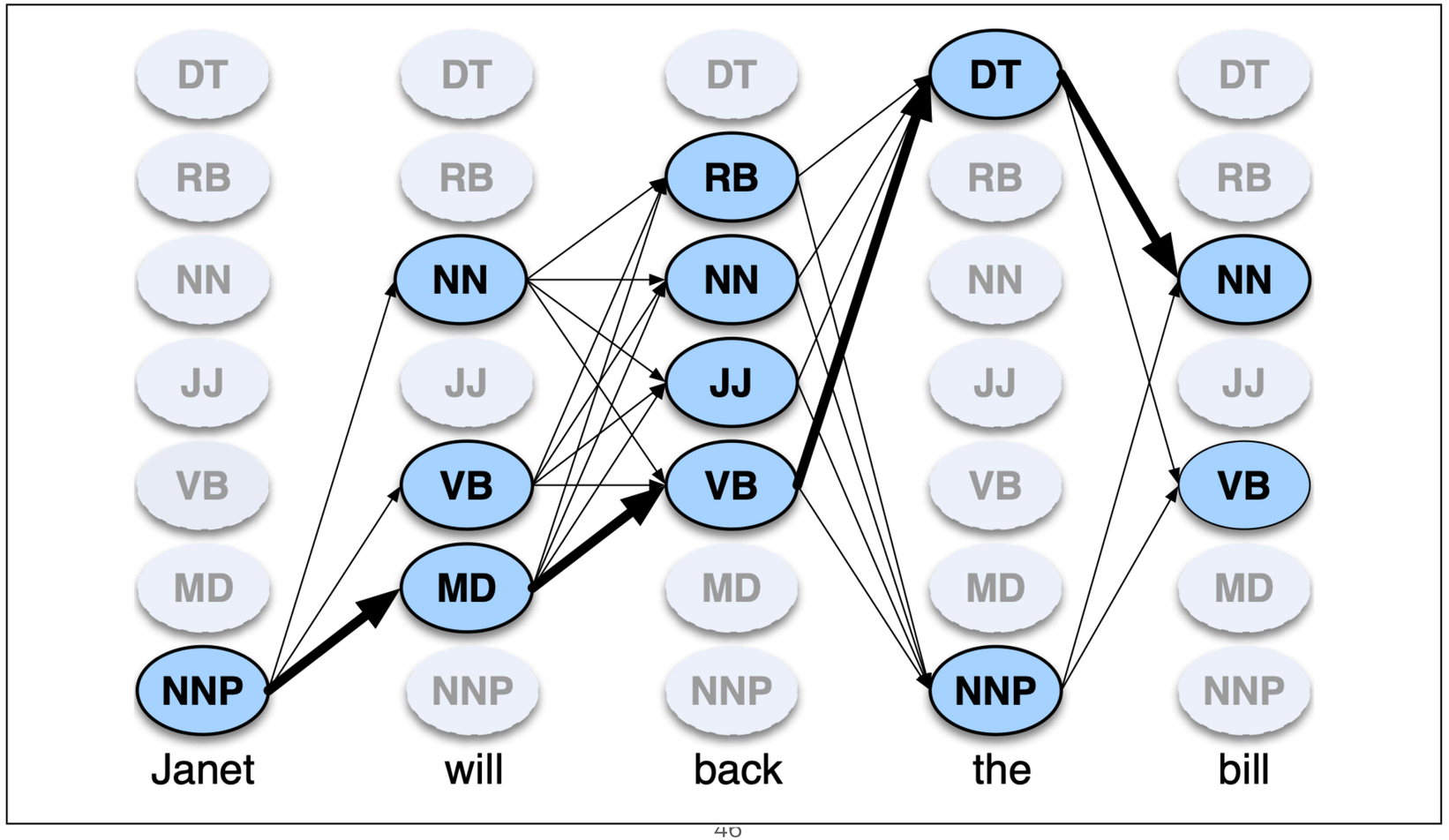
The Viterbi Algorithm

(1) Given an HMM containing hidden variables and an observation sequence the task of determining which sequence of variables is the underlying source of the observed sequence is called the **decoding** task.

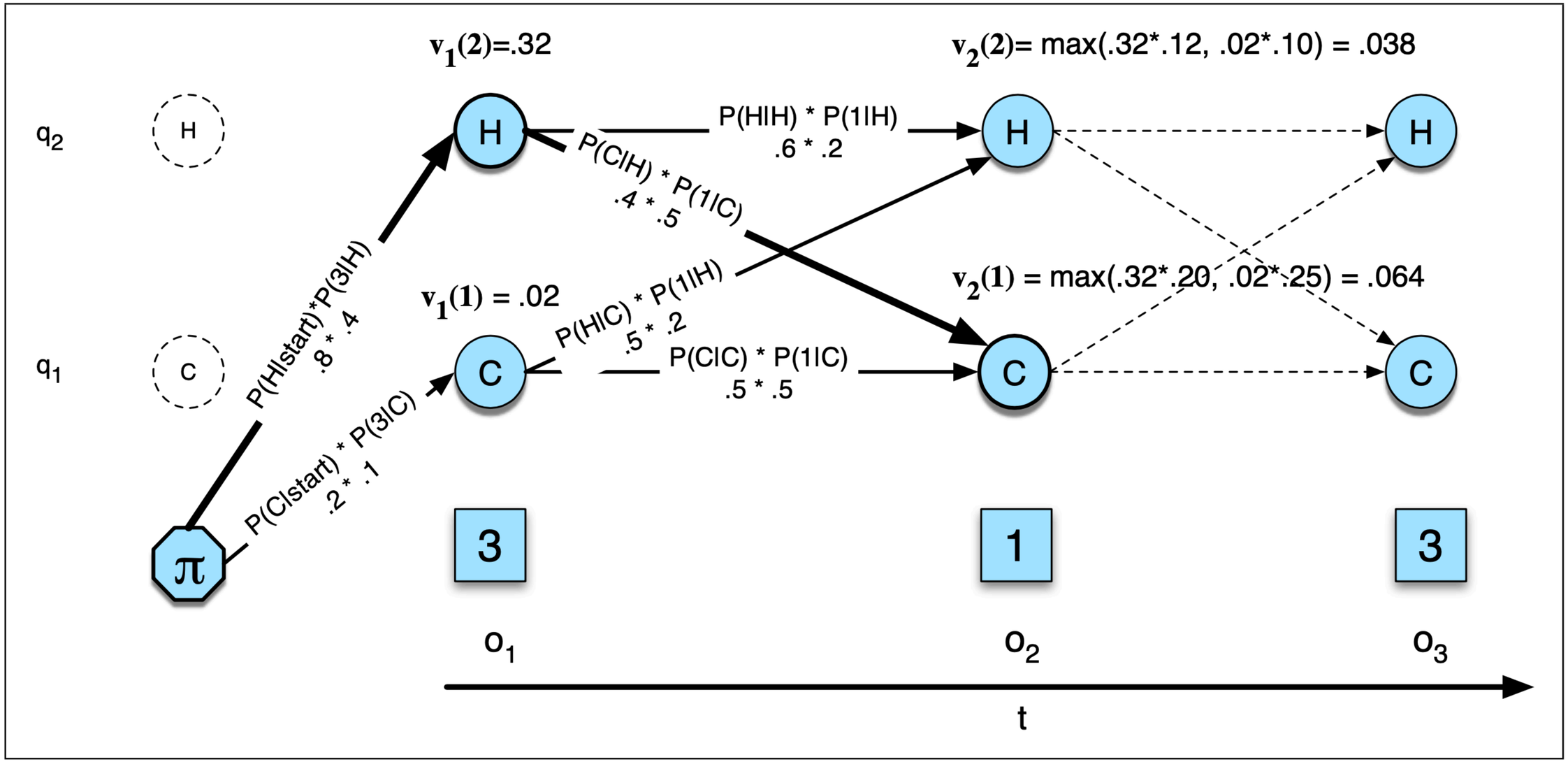
(2) The **Viterbi** algorithm is the most common algorithm used for decoding HMMs, both for POS tagging and for speech recognition.
- The Viterbi algorithm is an application of **dynamic programming**

(3) Given an HMM and an observation sequence, the Viterbi algorithm returns the state-path through the HMM which assigns maximum likelihood to the observation sequence.

HMM Decoding in POS-tagging



The Viterbi Algorithm – Example



The Viterbi Algorithm

(1) Notes:

a[s′,s] is the transition probability from the previous state s′ to the current state s

(2) $b_s(o_t)$ is the observation likelihood of o_t given s

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

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

CRF
Conditional Random Fields

Linear chain CRF

Which one is where (HMM / CRF)?

(1) X is input sequence, Y is sequience of tags

$$\begin{aligned}\hat{Y} &= \operatorname*{argmax}_{Y} p(Y|X) \\ &= \operatorname*{argmax}_{Y} p(X|Y)p(Y) \\ &= \operatorname*{argmax}_{Y} \prod_{i} p(x_i|y_i) \prod_{i} p(y_i|y_{i-1})\end{aligned}$$

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}} P(Y|X)$$

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

Linear chain CRF

Which one is where (HMM / CRF)?

(1) X is input sequence, Y is sequience of tags

CRF

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}} P(Y|X)$$

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

HMM

$$\begin{aligned}\hat{Y} &= \operatorname*{argmax}_{Y} p(Y|X) \\ &= \operatorname*{argmax}_{Y} p(X|Y)p(Y) \\ &= \operatorname*{argmax}_{Y} \prod_i p(x_i|y_i) \prod_i p(y_i|y_{i-1})\end{aligned}$$

What are F(X,Y)?

(1) Features:

$\mathbb{1}\{x_i = \textit{the}, y_i = \text{DET}\}$
$\mathbb{1}\{y_i = \text{PROPN}, x_{i+1} = \textit{Street}, y_{i-1} = \text{NUM}\}$
$\mathbb{1}\{y_i = \text{VERB}, y_{i-1} = \text{AUX}\}$

$$F_k(X, Y) = \sum_{i=1}^{K} f_k(y_{i-1}, y_i, X, i)$$

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

Feature Templates

$\langle y_i, x_i \rangle, \langle y_i, y_{i-1} \rangle, \langle y_i, x_{i-1}, x_{i+2} \rangle$

f_{3743}: y_i = VB and x_i = back
f_{156}: y_i = VB and y_{i-1} = MD
f_{99732}: y_i = VB and x_{i-1} = will and x_{i+2} = bill

x_i contains a particular prefix (perhaps from all prefixes of length ≤ 2)
x_i contains a particular suffix (perhaps from all suffixes of length ≤ 2)
x_i's word shape
x_i's short word shape

Inference and Training for CRFs

inference == decoding

(1) Can we use Viterbi for decoding?

$$
\begin{aligned}
\hat{Y} &= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} P(Y|X) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \exp\left(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)
\end{aligned}
$$

Homework

optional: apply Viterbi algorithm to find POS tags

(1) a bear likes sweet honey

(2) [DT] [ADJ, VERB, NOUN] [VERB, NOUN] [ADJ, NOUN] [ADJ, NOUN]

	ADJ	NOUN	VERB	DT
ADJ	0.4	0.4	0.2	0.1
NOUN	0.2	0.4	0.4	0.1
VERB	0.1	0.6	0.3	0.1
DT	0.29	0.6	0.1	0.01
[START]	0.1	0.4	0.1	0.4

	a	bear	likes	fly	honey	sweet
ADJ	0.2	0.1	0.1	0.1	0.1	0.4
NOUN	0.1	0.2	0.2	0.2	0.2	0.1
VERB	0.1	0.2	0.3	0.2	0.1	0.1
DT	0.5	0.1	0.05	0.1	0.1	0.1

Some source code examples (NLTK)

(1) https://www.mygreatlearning.com/blog/pos-tagging/#optimizing-hmm-with-viterbi-algorithm

HMM: A Toy Example

Happy/Sad and Colors

(1) Credits: https://www.youtube.com/watch?app=desktop&v=fX5bYmnHqqE&t=0s

Summary