

# PART-OF-SPEECH (POS) TAGGING

---

Slides adapted from Dan Jurafsky, Julia Hirschberg, Jim Martin, Chapter 8

**1. Introduction**

2. POS tagging approaches

3. Evaluation

## POS examples

- N            noun            chair, bandwidth, pacing
- V            verb            study, debate, munch
- ADJ        adjective        purple, tall, ridiculous
- ADV        adverb        unfortunately, slowly
- P            preposition    of, by, to
- PRO        pronoun        I, me, mine
- DET        determiner    the, a, that, those

## Penn TreeBank POS Tag set

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	<i>+, %, &amp;</i>
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VCN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WP\$	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	<i>\$</i>
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	<i>#</i>
PDT	Predeterminer	<i>all, both</i>	"	Left quote	<i>(' or ")</i>
POS	Possessive ending	<i>'s</i>	"	Right quote	<i>(' or ")</i>
PRP	Personal pronoun	<i>I, you, he</i>	(	Left parenthesis	<i>( [ , { , &lt;)</i>
PRP\$	Possessive pronoun	<i>your, one's</i>	)	Right parenthesis	<i>( ] , } , &gt;)</i>
RB	Adverb	<i>quickly, never</i>	,	Comma	<i>,</i>
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	<i>(. ! ?)</i>
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	<i>(: ; ... -)</i>
RP	Particle	<i>up, off</i>			

## Example of Penn Treebank Tagging of Brown Corpus Sentence

•VB DT NN .  
Book that flight .

•VBZ DT NN VB NN ?  
Does that flight serve dinner ?

•See <http://www.infogistics.com/posdemo.htm>

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential <i>there</i>
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRPS	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	<i>to</i>
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VCN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WPS	Possessive wh-pronoun
36.	WRB	Wh-adverb

## What is a word class?

- Words that somehow 'behave' alike:
  - Appear in similar contexts
  - Perform similar functions in sentences
  - Undergo similar transformations
- Called: **parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS**
- Basic word classes: 8 (ish) traditional parts of speech: *noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, ...*
- Modern works use extended lists of POS:
  - Brown corpus tagset (87 tags): <http://www.scs.leeds.ac.uk/amalgam/tagsets/brown.html>
  - Penn Treebank tagset (45 tags): <http://www.cs.colorado.edu/~martin/SLP/Figures/>
  - C7 tagset (146 tags): <http://www.comp.lancs.ac.uk/ucrel/claws7tags.html>

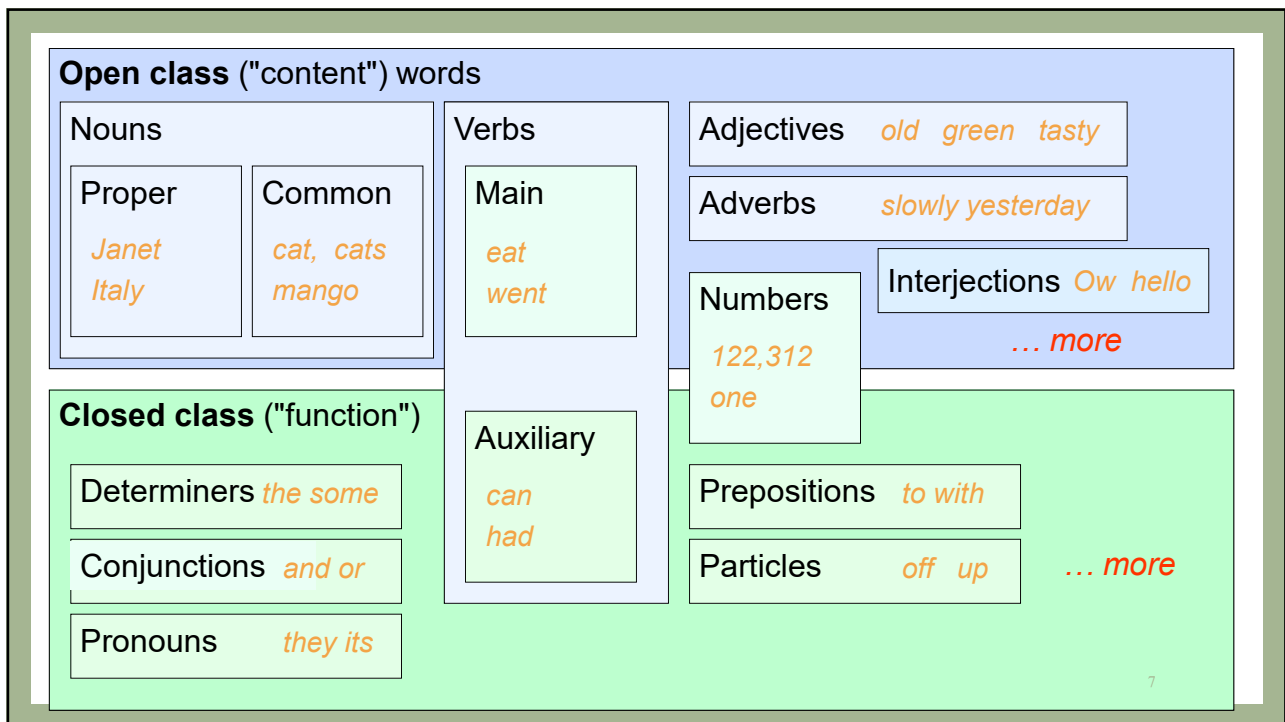
### 2 CLASSES OF WORDS

#### Open class

- Nouns, Verbs, Adjectives, Adverbs.
- Why "open"? → **new ones can be created all the time**
- English has 4: Nouns, Verbs, Adjectives, Adverbs
- Many languages have all 4, but not all!

#### Closed class (a relatively fixed membership )

- conjunctions: and, or, but
- pronouns: I, she, him
- prepositions: with, on, under, over, near, by, ...
- determiners: the, a, an
- Usually **function words** (short common words which play a role in grammar)



## POS tagging

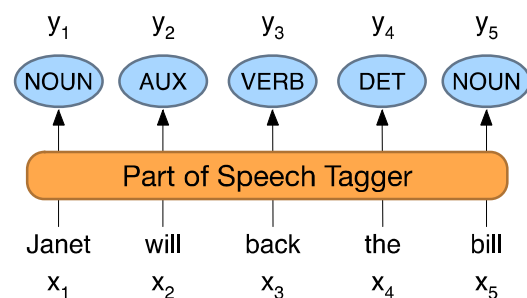
“The process of assigning a part-of-speech or other lexical class marker to each word in a corpus” (Jurafsky and Martin)

Map from sequence  $x_1, \dots, x_n$  of words to  $y_1, \dots, y_n$  of POS tags, each output  $y_i$  corresponding exactly to one input  $x_i$

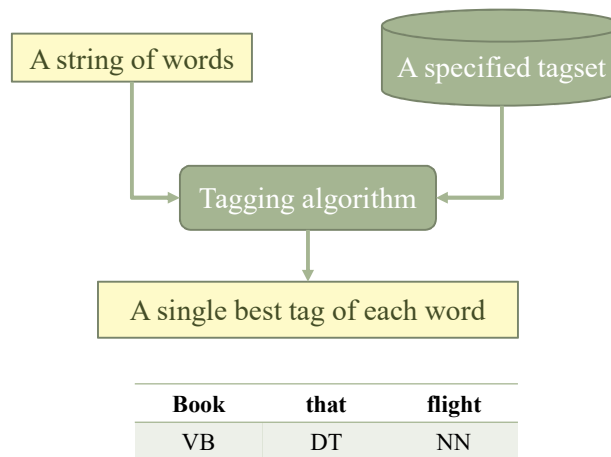
Tagging is a *disambiguation* task; words are *ambiguous* - have more than one possible part-of-speech

→ the goal is to **find** the correct **tag** for the situation

<https://web.stanford.edu/~jurafsky/slp3/8.pdf>



## Process of POS Tagging



LNK\_9

## 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** → PROP

LNK\_10

# POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential <i>there</i>
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRPS	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	<i>to</i>
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VCN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WPS	Possessive wh-pronoun
36.	WRB	Wh-adverb

## How do we assign POS tags to words in a sentence?

“Like” can be a verb or a preposition

- I like/**VB** candy.
- Time flies like/**IN** an arrow.

“Around” can be a preposition, particle, or adverb

- I bought it at the shop around/**IN** the corner.
- I never got around/**RP** to getting a car.
- A new Prius costs around/**RB** \$25K

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential <i>there</i>
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRPS	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	<i>to</i>
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VCN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WPS	Possessive wh-pronoun
36.	WRB	Wh-adverb

## How hard is POS tagging?

### Measuring ambiguity

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

## Defining an annotation scheme

- Training and evaluating models for these NLP tasks requires large corpora annotated with the desired representations.
- Annotation at scale is expensive, so a few existing corpora and their annotations and annotation schemes (tag sets, etc.) often become the de facto standard for the field.
- It is difficult to know what the “right” annotation scheme should be for any particular task
  - How difficult is it to achieve high accuracy for that annotation?
  - How useful is this annotation scheme for downstream tasks in the pipeline?
  - ⇨ We often can’t know the answer until we’ve annotated a lot of data...

1. Introduction

2. POS tagging  
approaches

3. Evaluation

15

## Algorithms for POS Tagging

$W = w_1 w_2 \dots w_n$  sequence of words

$T = t_1 t_2 \dots t_n$  sequence of POS tags

$f: W \rightarrow T = f(W)$

- Why can't we just look them up in a dictionary?
  - Words that aren't in the dictionary
- One idea:  $P(t_i | w_i)$  = the probability that a random hapax legomenon in the corpus has tag  $t_i$ .
  - Nouns are more likely than verbs, which are more likely than pronouns.
- Another idea: use morphology.

LNK\_16



## Algorithms for POS Tagging - Knowledge

- Dictionary
- Morphological rules, e.g.,
  - \_\_\_\_\_-tion
  - \_\_\_\_\_-ly
  - capitalization
- N-gram frequencies
  - to \_\_\_\_\_
  - DET \_\_\_\_\_ N
  - But what about rare words, e.g, *smelt* (two verb forms, smelt and past tense of smell, and one noun form, a small fish)
- Combining these
  - V \_\_\_\_\_-ing    *I was gracking* vs. *Gracking is fun.*

17

## Algorithms for POS Tagging - Approaches

- Basic approaches
  - Rule-Based
  - HMM-based
  - Transformation-Based Tagger (Brill) (we won't cover this)
- Do we return one best answer or several answers and let later steps decide?
- How does the requisite knowledge get entered?

18

- **Training/Teaching an NLP Component**

- Each step of NLP analysis requires a module that knows what to do. How do such modules get created?
  - By hand → advantage: based on sound linguistic principles, sensible to people, explainable
  - By training → less work, extensible to new languages, customizable for specific domains.

- **Training/Teaching a POS Tagger**

- The problem is tractable. We can do a very good job with just:
  - a dictionary
  - a tagset
  - a large corpus, usually tagged by hand

19

## **POS tagging approaches**

- **Rule-based POS tagging**
- Statistical POS tagging
- Markov chain
- Hidden Markov Model
- HMM POS tagging

LNK\_20

# Rule-based POS Tagging

- Basic Idea:
  - Start with a dictionary
  - Assign all possible tags to words from the dictionary
  - Write rules by hand to selectively remove tags
    - *if **word+1** is an adj, adv, or quantifier and the following is a sentence boundary and **word-1** is not a verb like “consider” then eliminate non-adv else eliminate adv.*
    - Typically more than 1000 hand-written rules
  - Leaving the correct tag for each word

21

## Rule-Based POS Tagging Start with a dictionary

- she: PRP
- promised: VBN,VBD
- to TO
- back: VB, JJ, RB, NN
- the: DT
- bill: NN, VB
- 
- Etc... for the ~100,000 words of English

## Rule-Based POS Tagging

Use the dictionary to assign every possible tag

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

## Rule-Based POS Tagging

Write rules to eliminate tags

Eliminate *VBN* if *VBD* is an option when *VBN|VBD* follows “<start> *PRP*”

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

## Sample ENGTWOL Lexicon

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
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	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

## Rule-Based POS Tagging ENGTWOL (ENGLISH TWO Level analysis)

- 1<sup>st</sup> stage: Run words through a morphological analyzer to get all parts of speech.
  - Example: *Pavlov had shown that salivation ...*

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

# Rule-Based POS Tagging ENGTWOL

- **2<sup>nd</sup> stage: Figure out what to do about words that are unknown or ambiguous.** Two approaches:
  - Rules that specify what to do.
  - Rules that specify what not to do:

## Adverbial-that rule

**Given input:** “that”

**If**

(+1 A/ADV/QUANT) ;if next word is adj/adv/quantifier

(+2 SENT-LIM) ; and following is a sentence boundary

(NOT -1 SVOC/A) ; and the previous word is not a verb like “consider” which allows adjective complements in “I consider that odd”

**Then** eliminate non-ADV tags

**Else** eliminate ADV

*It isn't that odd* vs

*I consider that odd* vs

*I believe that he is right.*

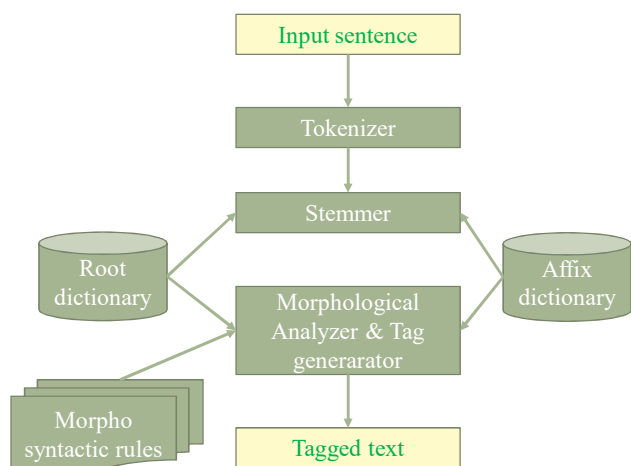
From ENGTWOL

27

## System Diagram of Rule based Morphology driven POS Tagger

### Algorithm

1. Give input text to the tokenizer module.
2. Repeat step 3 and 4 until each token is tagged.
3. Check for prefixes and suffixes and separate them with the help of affix dictionaries and check if the stemmed word occurs in the root dictionary or not. The words which are not stemmed are sent to the complex word handler module.
4. The complex words are stemmed separately, if these words are not stemmed by complex word handler and tag them as the Named Entities (NEs).
5. Apply the morphological rules on the affixes and root words for identifying the POS tag of the words according to the output of the morphological analyzer



Patra, Braja Gopal, Khumbar Debbarma, Dipankar Das, and Sivaji Bandyopadhyay. "Part of speech (pos) tagger for kokborok." In *Proceedings of COLING 2012: Posters*, pp. 923-932. 2012.

LNK\_28

## POS tagging approaches

- Rule-based POS tagging
- **Statistical POS tagging**
- Markov chain
- Hidden Markov Model
- HMM POS tagging

LNK\_29

## Most-frequent-tag

- **Most-frequent-tag algorithm**
  - For each word
    - Create dictionary with each possible tag for a word
    - Take a tagged corpus
    - Count the number of times each tag occurs for that word
  - Given a new sentence
    - For each word, pick the most frequent tag for that word from the corpus.

→ Q: Where does the dictionary come from?

A: One option is to use the same corpus that we use for computing the tags

## Statistical POS tagging

- Based on probability theory
- No probabilities for words not in corpus
- Conditional Probability and Tags
  - $P(Verb)$  is probability of randomly selected word being a verb.
  - $P(Verb|race)$  is “what’s the probability of a word being a verb given that it’s the word “race”?”
    - *Race* can be a **noun** or a **verb**. It’s more likely to be a noun
    - $P(Verb|race) =$  “out of all the times we saw ‘race’, how many were verbs?”
    - In Brown corpus,  $P(Verb|race) = 96/98 = .98$

$$P(V | race) = \frac{\text{Count}(race \text{ is verb})}{\text{total Count}(race)}$$

## Stochastic (Probabilistic) tagging

Based on probability of certain tag occurring, given various possibilities

- Necessitates a training corpus
  - A collection of sentences that have already been tagged
- Several such corpora exist
  - One of the best known is the Brown University Standard Corpus of Present-Day American English (or just the Brown Corpus)
  - about 1,000,000 words from a wide variety of sources
    - POS tags assigned to each



## Probabilistic tagging – approach #1

Assign each word its most likely POS tag

If  $w$  has tags  $t_1, \dots, t_k$ , then can use  $P(t_i|w) = \frac{c(w|t_i)}{c(w|t_1)+c(w|t_2)+\dots+c(w|t_k)}$ ,

where  $c(w|t_i)$  = number of times  $w|t_i$  appears in the corpus

Success: 91% for English

Example:

heat :: noun/89, verb/5

LNK\_33

## Probabilistic tagging – approach #2

Given: sequence of words (a sentence)  $W = w_1, w_2, \dots, w_n$

Assign sequence of tags  $T = t_1, t_2, \dots, t_n$

Find  $T$  that maximizes  $P(T|W)$

LNK\_34

## Probabilistic tagging – approach #2

Find  $T$  that maximizes  $P(T|W)$

By Bayes' rule:  $P(T|W) = \frac{P(W|T)P(T)}{P(W)} = \alpha P(W|T)P(T)$

So find  $T$  that maximizes  $P(W|T)P(T)$

- Chain rule:  $P(T) = P(t_1)P(t_2|t_1)P(t_3|t_1, t_2)P(t_4|t_1, t_2, t_3) \dots P(t_n|t_1, t_2, t_3 \dots t_{n-1})$
- As an approximation, use  $P(T) \approx P(t_1)P(t_2|t_1)P(t_3|t_2) \dots P(t_n|t_{n-1})$

*Assume each word is dependent only on its own POS tag: given its POS tag, it is conditionally independent of the other words around it.*

→ Then  $P(W|T) = P(w_1|t_1)P(w_2|t_2) \dots P(w_n|t_n)$

So  $P(T)P(W|T) \approx P(t_1)P(t_2|t_1) \dots P(t_n|t_{n-1})P(w_1|t_1)P(w_2|t_2) \dots P(w_n|t_n)$

LNK\_35

## Probabilistic tagging – approach #2

Find  $T$  that maximizes  $P(T|W)$

We want to compute

$$P(T)P(W|T) \approx P(t_1)P(t_2|t_1) \dots P(t_n|t_{n-1})P(w_1|t_1)P(w_2|t_2) \dots P(w_n|t_n)$$

Let

$$\begin{aligned} c(t_i) &= \text{frequency of } t_i \text{ in the corpus} \\ c(w_i, t_i) &= \text{frequency of } (w_i|t_i) \text{ in the corpus} \\ c(t_{i-1}, t_i) &= \text{frequency of } t_{i-1}t_i \text{ in the corpus} \end{aligned}$$

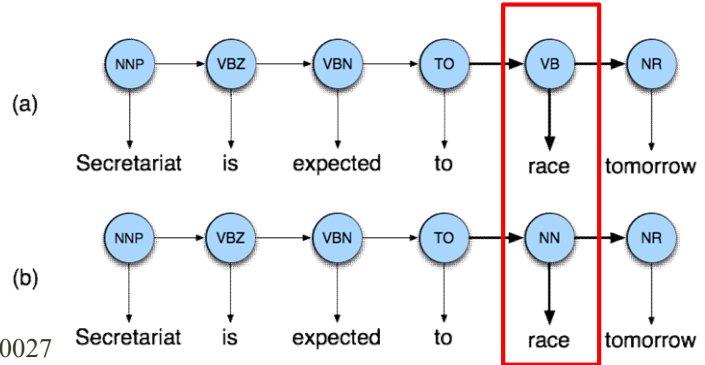
Then, we can use

$$\begin{aligned} P(t_i|t_{i-1}) &= \frac{c(t_{i-1}, t_i)}{c(t_{i-1})} \\ P(w_i|t_i) &= \frac{c(w_i, t_i)}{c(t_i)} \end{aligned}$$

LNK\_36

## Disambiguating “race”

- $P(NN|TO) = .00047$
- $P(VB|TO) = .83$
- $P(\text{race}|NN) = .00057$
- $P(\text{race}|VB) = .00012$
- $P(NR|VB) = .0027$
- $P(NR|NN) = .0012$
- $P(VB|TO)P(NR|VB)P(\text{race}|VB) = .00000027$
- $P(NN|TO)P(NR|NN)P(\text{race}|NN) = .0000000032$
- So  $\rightarrow ???$



37

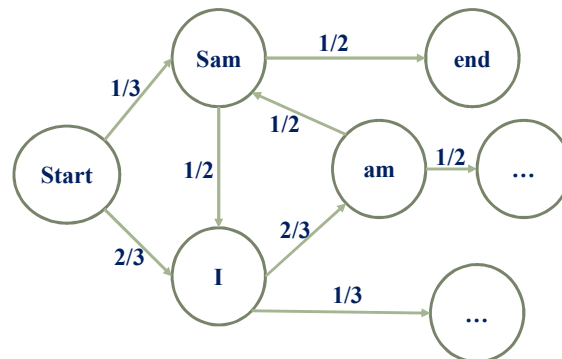
## POS tagging approaches

- Rule-based POS tagging
- Statistical POS tagging
- **Markov chain**
- Hidden Markov Model
- HMM POS tagging

LNK\_38

# Markov chain

- A Markov chain is a model that tells us something about the probabilities of sequences of random variables, **states**, each of which can take on values from some set.
  - These sets can be words, or tags, or symbols representing anything, like the weather
- Markov assumption: *if we want to predict the future state in the sequence, all that matters is the current state*



$$P(q_i|q_1 \dots q_{i-1}) = P(q_i|q_{i-1})$$

$$P(q_1, q_2, \dots, q_n) = \prod_{i=1}^n P(q_i|q_{i-1})$$

39

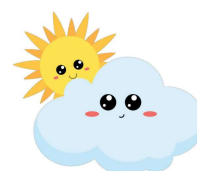
## Markov chain – Ví dụ

- Xác suất chuyển đổi thời tiết ngày hôm nay dựa trên ngày hôm qua như sau:

		Thời tiết ngày mai		
		Nắng	Mưa	Có_mây
Thời tiết hôm nay	Nắng	0,8	0,05	0,15
	Mưa	0,2	0,6	0,2
	Có_mây	0,2	0,3	0,5



Nắng  
(Sunny)



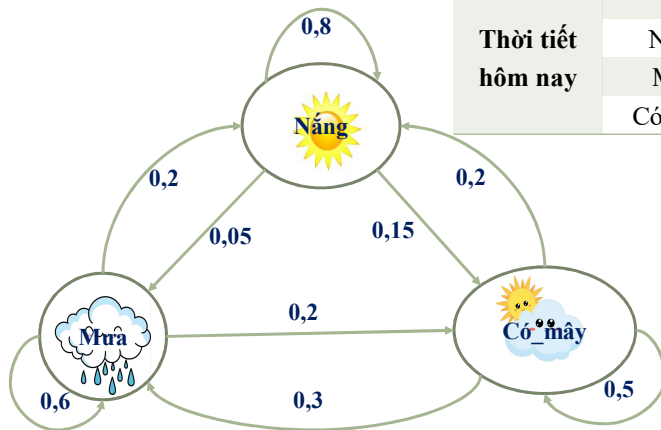
Có mây  
(Cloudy)



Mưa  
(Rainy)

40

## Markov chain – Ví dụ

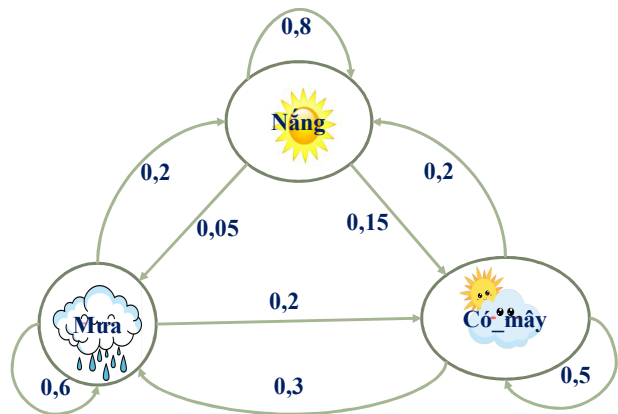


		Thời tiết ngày mai		
Thời tiết hôm nay		Nắng	Mưa	Có_mây
	Nắng	0,8	0,05	0,15
	Mưa	0,2	0,6	0,2
	Có_mây	0,2	0,3	0,5

41

## Markov chain – Ví dụ

- Nếu hôm qua thời tiết là “nắng”, và hôm nay là “nắng”, xác suất để ngày mai là “mưa” sẽ là:

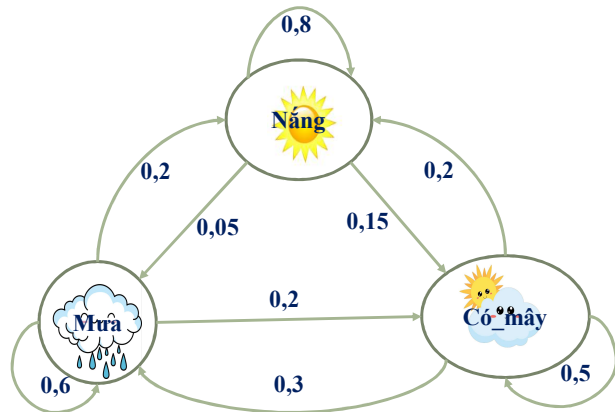


$$P(\text{mưa}|\text{nắng}, \text{nắng}) = P(\text{mưa}|\text{nắng}) = 0,05$$

42

## Markov chain – Ví dụ

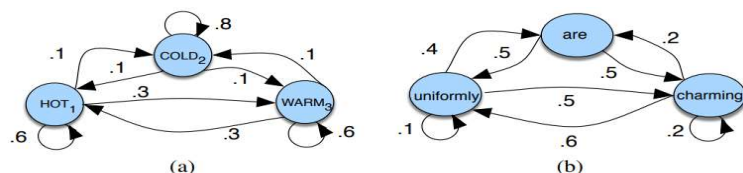
- Nếu hôm nay thời tiết là “có\_mây”, vậy xác suất để ngày mai là “có\_mây” và ngày mốt là “nắng” sẽ là:



$$\begin{aligned}
 &P(\text{có\_mây, nắng} | \text{có\_mây}) \\
 &= P(\text{có\_mây} | \text{có\_mây})P(\text{nắng} | \text{có\_mây, có\_mây}) \\
 &= P(\text{có\_mây} | \text{có\_mây})P(\text{nắng} | \text{có\_mây}) \\
 &= 0,5 * 0,2 = 0,1
 \end{aligned}$$

43

## Markov chain



- States:** nodes trong đồ thị
- Transitions:** edges nối các state với một giá trị xác suất chuyển trạng thái
  - Tổng xác suất của các transition đi ra từ 1 state phải bằng 1
- Thành phần của chuỗi Markov**

A Markov chain for weather (a) and one for words (b), showing states and transitions. A start distribution  $\pi$  is required; setting  $\pi = [0.1, 0.7, 0.2]$  for (a) would mean a probability 0.7 of starting in state 2 (cold), probability 0.1 of starting in state 1 (hot), etc.

$Q = q_1 q_2 \dots q_N$	Tập hợp $N$ trạng thái (state)
$A = a_{11} a_{12} \dots a_{nn}$	Ma trận xác suất chuyển trạng thái $A$ (transition probability matrix), mỗi phần tử $a_{ij}$ là xác suất chuyển từ trạng thái $i$ sang trạng thái $j$ , $\sum_{j=1}^n a_{ij} = 1, \forall i$
$\pi = \pi_1, \pi_2, \dots, \pi_N$	Phân phối xác suất khởi tạo (initial probability distribution), $\pi_i$ là xác suất để chuỗi Markov bắt đầu từ trạng thái $i$ . Nếu $\pi_i = 0$ có nghĩa là trạng thái $i$ không thể là trạng thái bắt đầu của chuỗi Markov. $\sum_{i=1}^n \pi_i = 1$ .

44

# Markov chain

- *Summary:* A Markov chain is a weighted automaton in which
  - weights are probabilities, i.e., all weights are between 0 and 1 and the sum of the weights of all outgoing edges of a state is 1, and
  - the input sequence uniquely determines the states the automaton goes through.
- A Markov chain is actually a bigram language model
- Markov chains are useful when we want to compute the probability for a sequence of events that we can observe.

LNK\_45

# Markov Model

- A Markov Model is a stochastic model which models temporal or sequential data, i.e., data that are ordered
- It provides a way to model the dependencies of current information (e.g. weather) with previous information
- It is composed of states, transition scheme between states, and emission of outputs (discrete or continuous)
- Several goals can be accomplished by using Markov models:
  - Learn statistics of sequential data.
  - Do prediction or estimation.
  - Recognize patterns

46

## POS tagging approaches

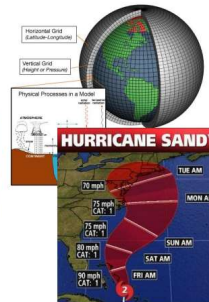
- Rule-based POS tagging
- Statistical POS tagging
- Markov chain
- **Hidden Markov Model**
- HMM POS tagging

### Speech Recognition and Translation



[https://chagall.med.cornell.edu/BioinfoCourse/presentations2013/Lecture3\\_HMM\\_2013.pdf](https://chagall.med.cornell.edu/BioinfoCourse/presentations2013/Lecture3_HMM_2013.pdf)

### Weather Modeling



### Sequence Alignment

```

RSKAFSPSHLQCHRRQIOETHEHNOQFAFFT 60
DQAFAGHSSLRKTHITHGEFYECHQCFASF 40
*****
FKKCSLLQRHKITHGEKPYE-CNQCGFAFAQ- 116
FSQHGLLRHKITHGEKPYMNVINAVPLHNS 98
*****

```

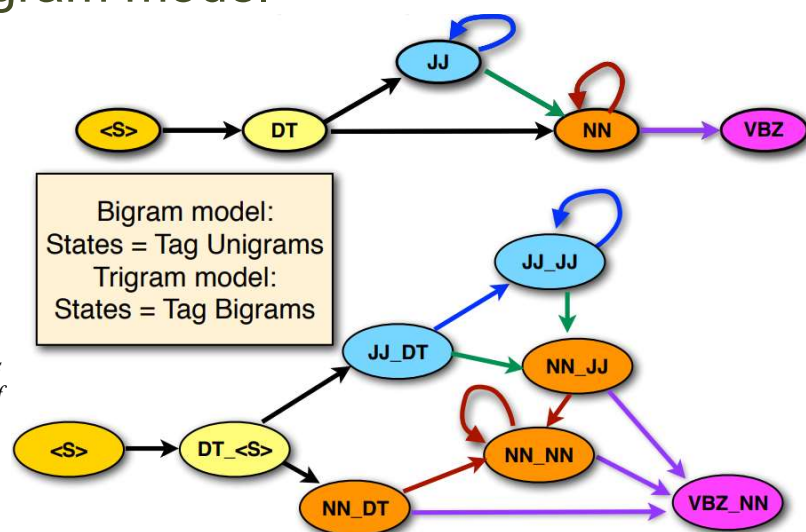
### Financial Modeling



HMM is useful for many problems including  
**POS Tagging**, Speech Recognition,  
 Weather Modeling, Sequence Alignment

LNK\_47

## Encoding a trigram model



<https://courses.grainger.illinois.edu/cs447/fa2019/Slides/Lecture10.pdf>



# Hidden Markov Model

- Bạn muốn biết thời tiết ngoài trời như thế nào....
- Toàn bộ thông tin bạn có là bạn quan sát thấy người giao hàng có mang dù (umbrella) hay không...



Nắng  
(Sunny)



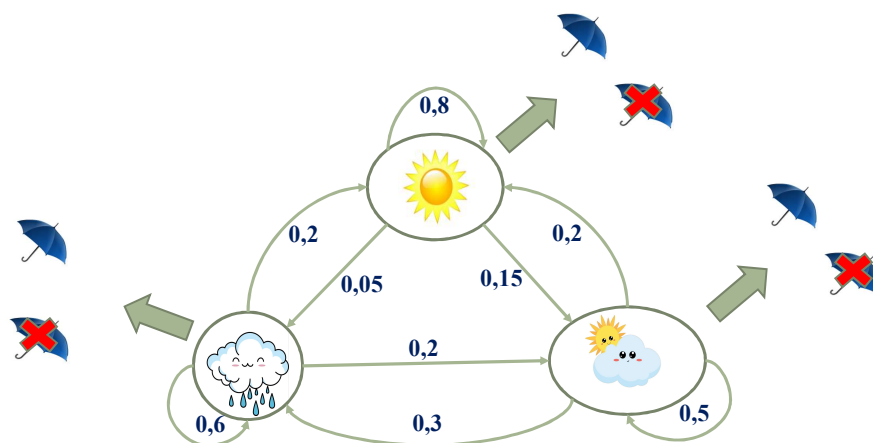
Có mây  
(Cloudy)



Mưa  
(Rainy)

49

# Hidden Markov Model



50

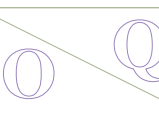

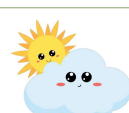
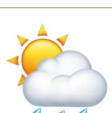


# HMM components and parameters

$Q = q_1 q_2 \dots q_N$	Tập hợp $N$ trạng thái ( <i>state</i> )
$A = a_{11} a_{12} \dots a_{nn}$	Ma trận xác suất chuyển trạng thái $A$ ( <i>transition probability matrix</i> ), mỗi phần tử $a_{ij}$ là xác suất chuyển từ trạng thái $i$ sang trạng thái $j$ , $\sum_{j=1}^n a_{ij} = 1, \forall i$
$O = \{o_1, o_2, \dots, o_t\}$	Chuỗi quan sát được ( <i>observation</i> ), mỗi observation được lấy từ bộ từ vựng $V = v_1, v_2, \dots, v_V$
$B = b_i(o_t)$	Chuỗi <i>observation likelihoods</i> , hay còn gọi là <i>emission probabilities</i> . Mỗi giá trị $b_i(o_t)$ là xác suất để observation $o_t$ được tạo thành từ trạng thái $q_i$
$\pi = \pi_1, \pi_2, \dots, \pi_N$	Phân phối xác suất khởi tạo ( <i>initial probability distribution</i> ), $\pi_i$ là xác suất để chuỗi Markov bắt đầu từ trạng thái $i$ . Nếu $\pi_i = 0$ có nghĩa là trạng thái $i$ không thể là trạng thái bắt đầu của chuỗi Markov. $\sum_{i=1}^n \pi_i = 1$ .

- HMM được điều chỉnh bởi 3 tham số:  $\lambda = \{A, B, \pi\}$

51

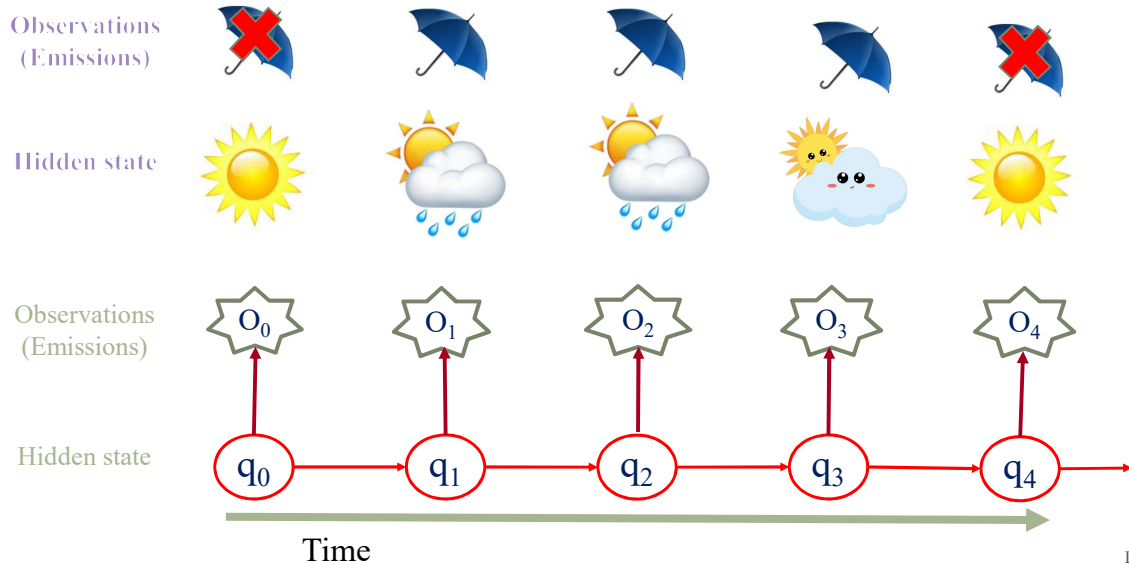
## Probabilistic reasoning

		Nắng	Có mây	Mưa
Observations				
		$P(\text{nắng} \text{có\_dù})$	$P(\text{có\_mây} \text{có\_dù})$	$P(\text{mưa} \text{có\_dù})$
		$P(\text{nắng} \text{không\_dù})$	$P(\text{có\_mây} \text{không\_dù})$	$P(\text{mưa} \text{không\_dù})$

$P(Q|O) = \text{probability of } Q \text{ happening if } O \text{ is observed}$

52

## Probabilistic reasoning in stochastic processes



LNK\_53

## Assumptions in Markov modeling

- Assumption 1: the current state depends only on the previous state

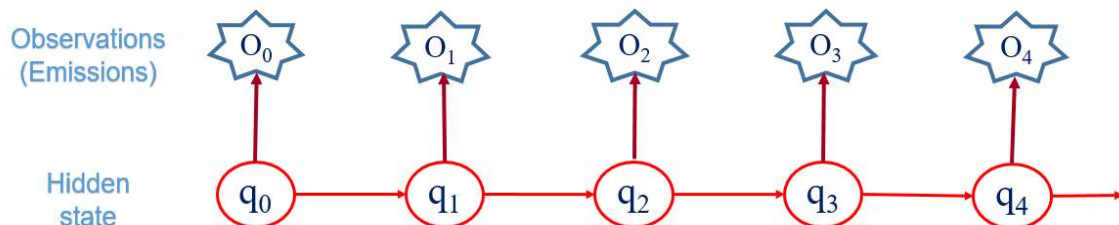
$$P(q_i | q_1 q_2 \dots q_{i-1}) = P(q_i | q_{i-1})$$

Transition probability

- Assumption 2: the current observation  $o_i$  depends only on the current state  $q_i$

$$P(o_i | q_1 \dots q_i \dots q_T, o_1 \dots o_i \dots o_T) = P(o_i | q_i)$$

Emission probability



# HMM

- Observation sequence  $O = \{o_1, \dots, o_i, \dots, o_T\}$ ,  $o_i \in \{\text{có\_dù}, \text{không\_dù}\}$ .
- Unknown sequence  $Q = \{q_1, \dots, q_i, \dots, q_T\}$ ,  $q_i \in \{\text{nắng}, \text{mưa}, \text{có\_mây}\}$ .
- $P(q_1, \dots, q_t | o_1, \dots, o_t) = ?$

55

# HMM

- From Bayes' Theorem, we can obtain the probability for a particular day as:  $P(q_i | o_i) = \frac{P(o_i | q_i)P(q_i)}{P(o_i)}$

- For a sequence of length  $t$ :

$$P(q_1, \dots, q_t | o_1, \dots, o_t) = \frac{P(o_1, \dots, o_t | q_1, \dots, q_t)P(q_1, \dots, q_t)}{P(o_1, \dots, o_t)}$$

- From the Markov property:

$$P(q_1, \dots, q_t) = \prod_{i=1}^t P(q_i | q_{i-1})$$

- Independent observations assumption:

$$P(o_1, \dots, o_t | q_1, \dots, q_t) = \prod_{i=1}^t P(o_i | q_i)$$

- Thus  $P(q_1, \dots, q_t | o_1, \dots, o_t) \propto \underbrace{\prod_{i=1}^t P(o_i | q_i) \prod_{i=1}^t P(q_i | q_{i-1})}_{\text{HMM Parameters}}$

- HMM Parameters:**
- Transition probabilities  $P(q_i | q_{i-1})$
  - Emission probabilities  $P(o_i | q_i)$
  - Initial state probabilities  $P(q_i)$

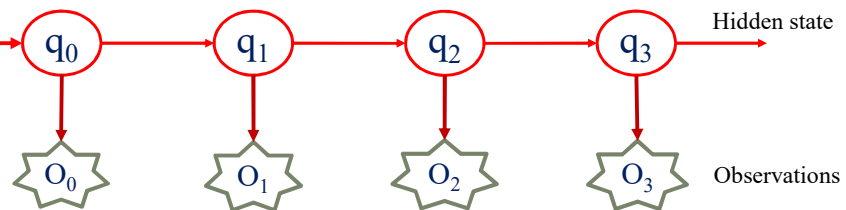
56

The initial and transition probability models:  $\pi$  and A

$\pi$   
Initial probability distribution

	$P(X)$
	$\pi$
Nắng	0,7
Mưa	0,15
Có_mây	0,15

Matrix A			
Transition probability matrix		$P(X_t X_{t-1})$	
$X_{t-1}$	$P(X_t=\text{nắng})$	$P(X_t=\text{mưa})$	$P(X_t=\text{có\_mây})$
Nắng	0,8	0,05	0,15
Mưa	0,2	0,6	0,2
Có_mây	0,2	0,3	0,5

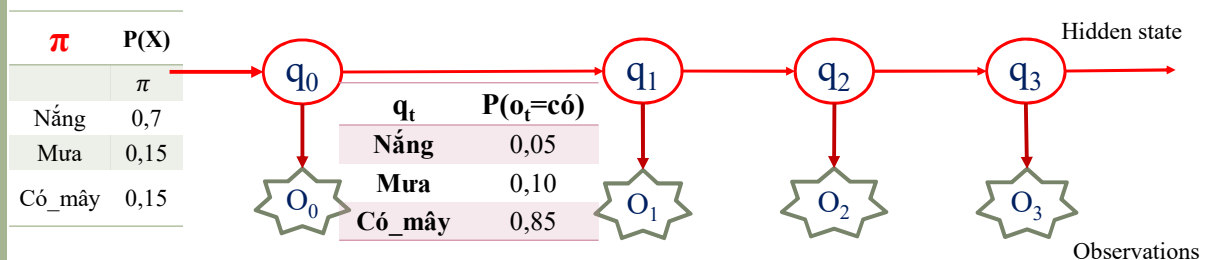


Encodes prior knowledge about the weather trends

57

The observation probability model: B

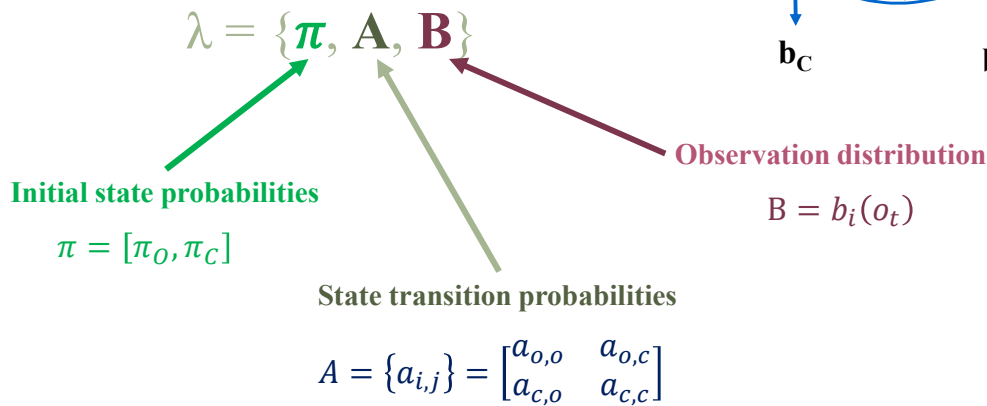
Matrix A			
$P(X_t X_{t-1})$		$P(X_t X_{t-1})$	
$X_{t-1}$	$P(X_t=\text{nắng})$	$P(X_t=\text{mưa})$	$P(X_t=\text{có\_mây})$
Nắng	0,8	0,05	0,15
Mưa	0,2	0,6	0,2
Có_mây	0,2	0,3	0,5



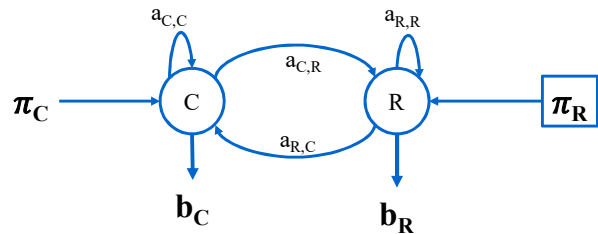
Encodes prior knowledge about how likely people are to bring their umbrella depending on weather conditions

58

## HMM parameters



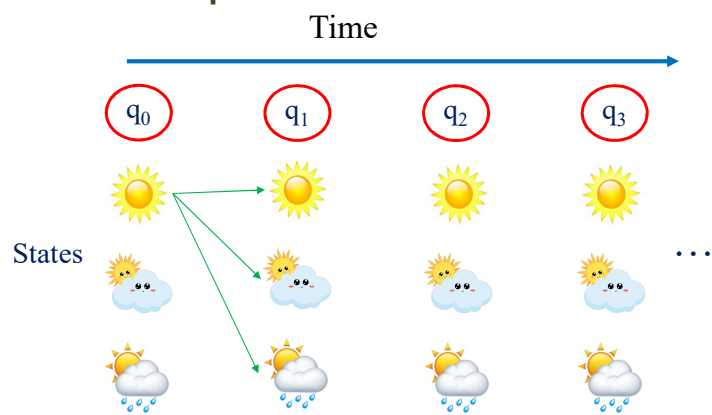
## Markov model



LNK\_59

## Finding the optimal state sequence with Viterbi

- Given a model that describes the system  $\lambda = \{\pi, \mathbf{A}, \mathbf{B}\}$ , we can determine the optimal state sequence (idealization) as follows:
- For each state at time  $t$ , calculate probability of the state at time  $t$  ( $q_t$ ) being a particular state  $q_i$  (“nắng”, “mưa”, “có mây”), given observations and previous states:



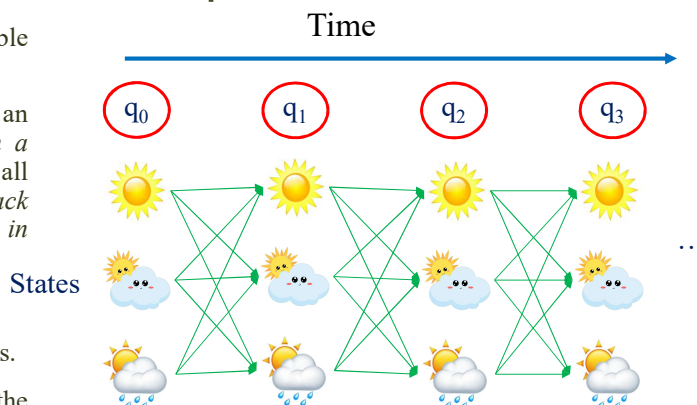
$$P(q_t | o_t, o_{t-1}, o_{t-2}, \dots, q_{t-1}, q_{t-2}, \dots) = P(q_t | o_t, q_{t-1}) = P(q_t | o_t) * P(q_t | q_{t-1})$$

## Finding the optimal state sequence with Viterbi

- Repeat these calculations for all possible transitions recursively.
- Then at each point in time we have an *estimate of how likely we are to be in a particular state at that time given all possible previous paths. We also keep track of the most likely state at each point in time.*

This complex looking thing is called a trellis.

- Find the **most likely end state** from the probabilities.
- We can then **backtrack** to find the most likely state sequence.



61

## HMM tagger

- The goal of HMM decoding is to choose the tag sequence  $t$  that is most probable given the observation sequence of  $n$  words

### HMM DECODING

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

Applying the Bayes rule  

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

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

HMM assumption

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

Argmax does not depend on the denominator, thus can be omitted

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

Bigram assumption

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

PUTTING ALL TOGETHER

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n \underbrace{P(w_i | t_i)}_{\text{emission}} \underbrace{P(t_i | t_{i-1})}_{\text{transition}}$$

62

# HMM- the three fundamental problems

- Problem 1. Likelihood of the input  $O$ :
  - Compute  $P(O|\lambda)$  for the input  $O$  and HMM  $\lambda$
  - → [Forward algorithm](#)
- Problem 2. Decoding (= *tagging*) the input  $O$ :
  - Find the best (*tags*)  $Q$  for the input  $O$
  - → [Viterbi](#)
- Problem 3. Estimation (= learning the model):
  - Find the best model parameters  $A$  and  $B$  for the training data  $O$
  - → [Forward-backward algorithm](#)

**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  $B$ .

63

## POS tagging approaches

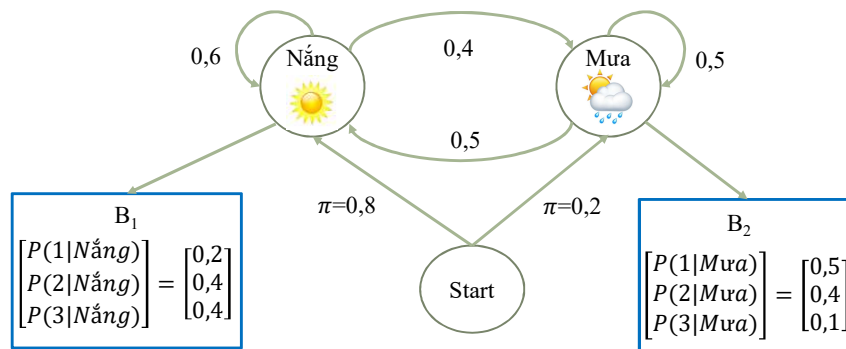
- Rule-based POS tagging
- Statistical POS tagging
- Markov chain
- **Hidden Markov Model**
  - The forward algorithm
  - The Viterbi algorithm
  - The forward-backward algorithm
- HMM POS tagging

LNK\_64



## Bài toán Ice-cream

- 2 trạng thái ẩn (hidden state) là “Nắng” và “Mưa”
- Chuỗi quan sát được *observations*  $O = \{1, 2, 3\}$  tương ứng với số kem ăn trong một ngày

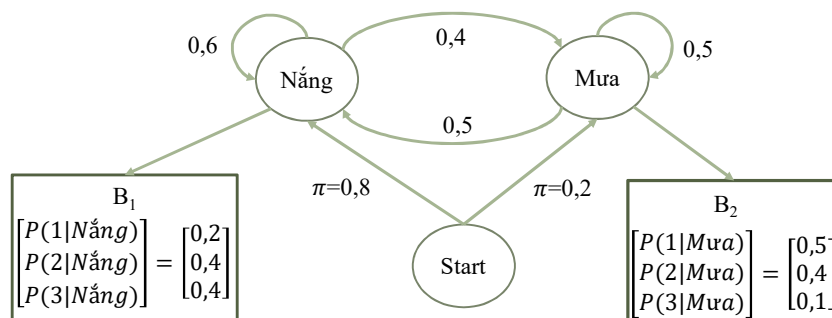


65

## HMM- Problem 1

- 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  $B$ .

- **Computing Likelihood:** Given an HMM  $\lambda = (A, B)$  and an observation sequence  $O$ , determine the likelihood  $P(O|\lambda)$ .
- **Example:** given the ice-cream eating HMM, what is the probability of the sequence 3 1 3?
  - → Problem: we don't know what the hidden state sequence is

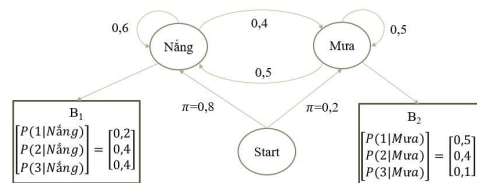


66

# HMM- Problem 1

Example: given the ice-cream eating HMM, what is the probability of the sequence 3 1 3?

Note: we don't know what the hidden state sequence is.



- In HMM, *each hidden state produces only a single observation* → sequence of hidden states and the sequence of observations have the same length
- Given:
  - A particular hidden state sequence  $Q = q_1, q_2, \dots, q_T$
  - An observation sequence  $O = o_1, o_2, \dots, o_T$
- Then, the likelihood of the observation sequence is  $P(O|Q) = \prod_{i=1}^T P(o_i|q_i)$
- We don't know what the hidden state sequence was → The joint probability of being in a particular weather sequence  $Q$  and generating a particular sequence  $O$

$$P(O, Q) = P(O|Q) * P(Q) = \prod_{i=1}^T P(o_i|q_i) * \prod_{i=1}^T P(q_i|q_{i-1})$$

- The total probability of the observations

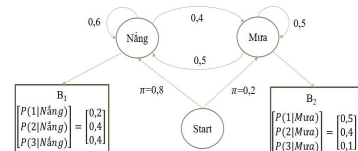
$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)$$

67

# HMM- Problem 1

Example: given the ice-cream eating HMM, what is the probability of the sequence 3 1 3?

Note: we don't know what the hidden state sequence is.



- Example: The computation of the forward probability for our ice-cream observation 3 1 3 from one possible hidden state sequence *nắng nắng mưa*

$$P(3 \ 1 \ 3 | \text{nắng nắng mưa}) = P(3|\text{nắng}) * P(1|\text{nắng}) * P(3|\text{mưa})$$

- We don't know what the hidden state sequence was → The sequence 3 1 3 has eight 3-event sequence

*nắng nắng mưa      mưa mưa nắng      mưa nắng mưa      nắng mưa mưa      ....*

$$P(3 \ 1 \ 3) = P(3 \ 1 \ 3, \text{nắng nắng mưa}) * P(3 \ 1 \ 3, \text{mưa mưa nắng}) * \dots$$

$$P(O, Q) = P(O|Q) * P(Q) = \prod_{i=1}^T P(o_i|q_i) * \prod_{i=1}^T P(q_i|q_{i-1})$$

$$P(3 \ 1 \ 3, \text{nắng nắng mưa}) = P(\text{nắng}|\text{start}) * P(\text{nắng}|\text{nắng}) * P(\text{mưa}|\text{nắng}) * P(3|\text{nắng}) * P(1|\text{nắng}) * P(3|\text{mưa})$$

- Problem: **N hidden states** and an observation sequence of **T observations**, there are **N<sup>T</sup> possible hidden sequences** → Solution: the forward algorithm

68

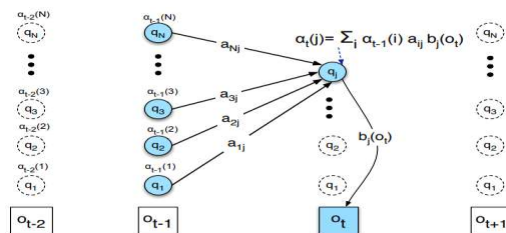
# The forward algorithm

- It is an efficient  $O(N^2T)$  algorithm and a kind of dynamic programming algorithm that uses a table to store intermediate values as it builds up the probability of the observation sequence.
- The forward algorithm **computes the observation probability by summing over the probabilities of all possible hidden state paths** that could generate the observation sequence.
- Each cell of the trellis  $\alpha_t(j)$  represents the probability of being in state  $j$  after seeing the first  $t$  observations, given the automaton  $\lambda$
- Given state  $q_j$  at time  $t$ , the value is computed as:  $\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t)$

$\alpha_{t-1}(i)$	the <b>previous forward path probability</b> from the previous time step
$a_{ij}$	the <b>transition probability</b> from previous state $q_i$ to current state $q_j$
$b_j(o_t)$	the <b>state observation likelihood</b> of the observation symbol $o_t$ given the current state $j$

69

## The forward algorithm



Visualizing the computation of a single element  $\alpha_t(i)$  in the trellis by summing all the previous values  $\alpha_{t-1}$ , weighted by their transition probabilities  $a$ , and multiplying by the observation probability  $b_j(o_t)$ .

1. Initialization:

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

2. Recursion:

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

3. Termination:

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

**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

The forward algorithm, where **forward[s,t]** represents  $\alpha_t(s)$ .

## The forward algorithm

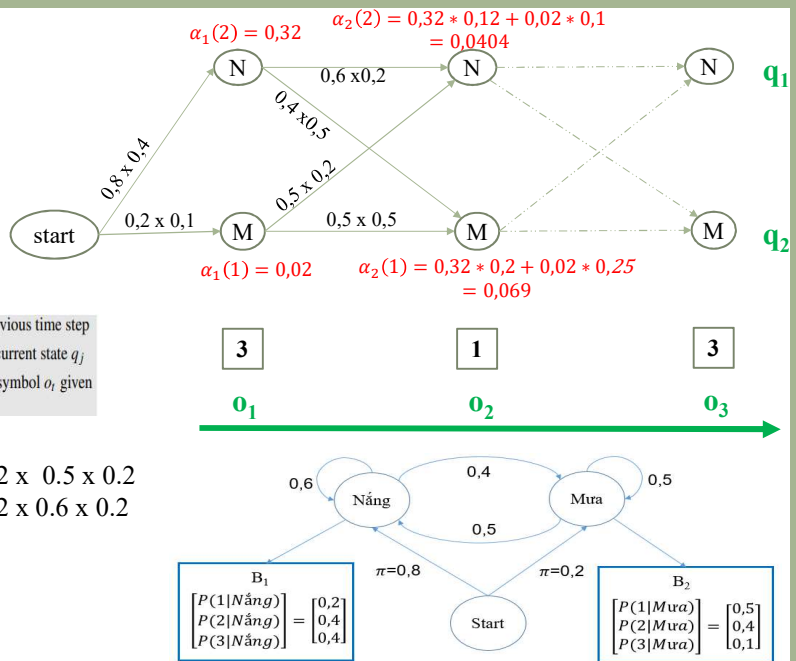
Given state  $q_j$  at time  $t$ , the value is computed as:

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

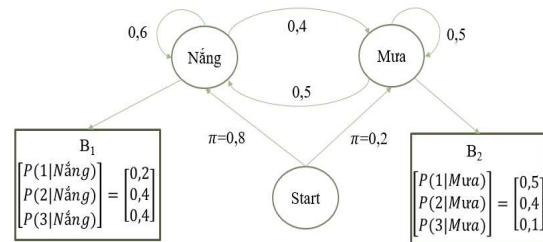
$\alpha_{t-1}(i)$  the previous forward path probability from the previous time step  
 $a_{ij}$  the transition probability from previous state  $q_i$  to current state  $q_j$   
 $b_j(o_t)$  the state observation likelihood of the observation symbol  $o_t$  given the current state  $j$

$$\alpha_2(2) = \alpha_1(1) \times P(N|N) \times P(1|N) = 0.02 \times 0.5 \times 0.2$$

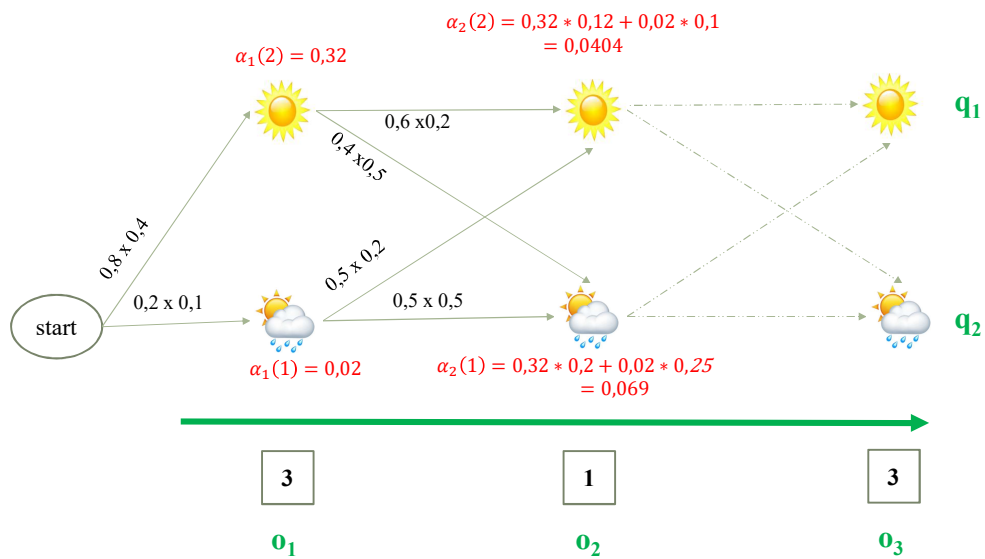
$$\alpha_2(2) = \alpha_1(2) \times P(N|N) \times P(1|N) = 0.32 \times 0.6 \times 0.2$$



## The forward algorithm

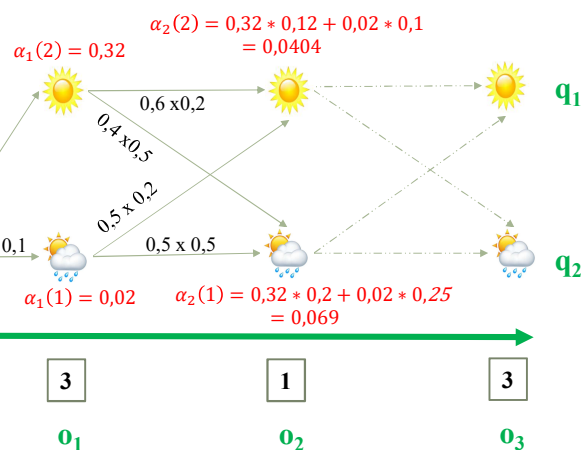
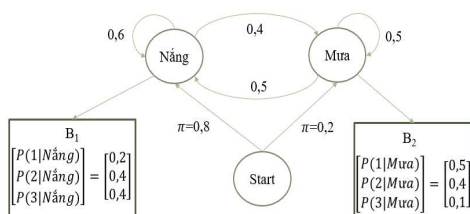
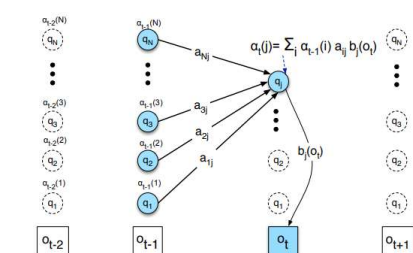


	3	1	3
<b>Mưa</b>	$\alpha_1(1) = P(M \text{start}) P(3 M) = 0.2 \times 0.1 = 0.002$	$\alpha_2(1) = 0.005 + 0.064 = 0.069$	$\alpha_3(1) = 0.00345 + 0.001616 = 0.005066$
		$\alpha_2(1) = \alpha_1(1) P(M M) P(1 M) = 0.002 \times 0.5 \times 0.5 = 0.0005$	$\alpha_3(1) = \alpha_2(1) P(M M) P(3 M) = 0.0069 \times 0.5 \times 0.1 = 0.000345$
		$\alpha_2(1) = \alpha_1(2) P(M N) P(1 M) = 0.32 \times 0.4 \times 0.5 = 0.064$	$\alpha_3(1) = \alpha_2(2) P(M N) P(3 M) = 0.0404 \times 0.4 \times 0.1 = 0.001616$
<b>Nắng</b>	$\alpha_1(2) = P(N \text{start}) P(3 N) = 0.8 \times 0.4 = 0.32$	$\alpha_2(2) = 0.002 + 0.0384 = 0.0404$	$\alpha_3(2) = 0.01104 + 0.009696$
		$\alpha_2(2) = \alpha_1(1) P(N M) P(1 N) = 0.02 \times 0.5 \times 0.2 = 0.002$	$\alpha_3(2) = \alpha_2(1) P(M N) P(3 N) = 0.069 \times 0.4 \times 0.4 = 0.01104$
		$\alpha_2(2) = \alpha_1(2) P(N N) P(1 N) = 0.32 \times 0.6 \times 0.2 = 0.0384$	$\alpha_3(2) = \alpha_2(2) P(N N) P(3 N) = 0.0404 \times 0.6 \times 0.4 = 0.009696$



73

## The forward algorithm



74

## The forward algorithm

- For each possible hidden state sequence, we could run the forward algorithm and compute the likelihood of the observation sequence given that hidden state sequence.
- Then we could choose the hidden state sequence with the maximum observation likelihood.
- It should be clear from the previous section that we cannot do this because there are an exponentially large number of state sequences.

75

## POS tagging approaches

- Rule-based POS tagging
- Statistical POS tagging
- Markov chain
- **Hidden Markov Model**
  - The forward algorithm
  - **The Viterbi algorithm**
  - The forward-backward algorithm
- HMM POS tagging

LNK\_76

# HMM- Problem 2

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

- For any model, such as an HMM, that contains hidden variables, the task of determining which sequence of variables is the underlying source of some sequence of observations is called the decoding task.
- In the ice-cream domain, given a sequence of ice-cream observations 3 / 3 and an HMM, the task of the decoder is to find the best hidden weather sequence (H H H)

**Decoding:** Given as input an HMM  $\lambda = (A, B)$  and a sequence of observations  $O = o_1, o_2, \dots, o_T$ , find the most probable sequence of states  $Q = q_1 q_2 q_3 \dots q_T$ .

- Viterbi algorithm
  - is a kind of dynamic programming that makes uses of a dynamic programming trellis
  - strongly resembles another dynamic programming variant, the minimum edit distance algorithm
  - The goal is to find the most likely sequence of hidden tags**

77

## The Viterbi algorithm

**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
         $viterbi[s, t] \leftarrow \max_{j=1}^N [viterbi[j, t-1] * a_{j,s} * b_s(o_t)]$ 
         $backpointer[s, t] \leftarrow \operatorname{argmax}_{j=1}^N [viterbi[j, t-1] * a_{j,s} * b_s(o_t)]$ 
     $bestpathprob \leftarrow \max_{s=1}^N viterbi[s, T]$  ; termination step
     $bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s, T]$  ; termination step
     $bestpath \leftarrow$  the path starting at state  $bestpathpointer$ , that follows  $backpointer[]$  to states back in time
return  $bestpath, bestpathprob$ 
    
```

Viterbi algorithm for finding optimal sequence of hidden states. Given an observation sequence and an HMM  $\lambda = (A, B)$ , the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

### 1. Initialization:

$$v_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

$$bt_1(j) = 0 \quad 1 \leq j \leq N$$

### 2. Recursion

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

$$bt_t(j) = \operatorname{argmax}_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

### 3. Termination:

$$\text{The best score: } P^* = \max_{i=1}^N v_T(i)$$

$$\text{The start of backtrace: } q_T^* = \operatorname{argmax}_{i=1}^N v_T(i)$$

78

# The Viterbi algorithm

- Given:
  - A particular hidden state sequence  $Q = q_0, q_1, q_2, \dots, q_T$
  - An observation sequence  $O = o_1, o_2, \dots, o_T$ ,
- Each cell of the trellis,  $v_t(j)$ , represents the probability that the HMM is in state  $j$  after seeing the first  $t$  observations and passing through the most probable state sequence  $q_1, \dots, q_{t-1}$ , given the automaton  $\lambda$
- The value of each cell  $v_t(j)$  is computed by recursively taking the most probable path that could lead us to this cell

$$v_t(j) = \max_{q_1, \dots, q_{t-1}} P(q_1 \dots q_{t-1}, o_1, o_2 \dots o_t, q_t = j | \lambda) \quad v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t)$$

$v_{t-1}(i)$  the **previous Viterbi path probability** from the previous time step  
 $a_{ij}$  the **transition probability** from previous state  $q_i$  to current state  $q_j$   
 $b_j(o_t)$  the **state observation likelihood** of the observation symbol  $o_t$  given the current state  $j$

79

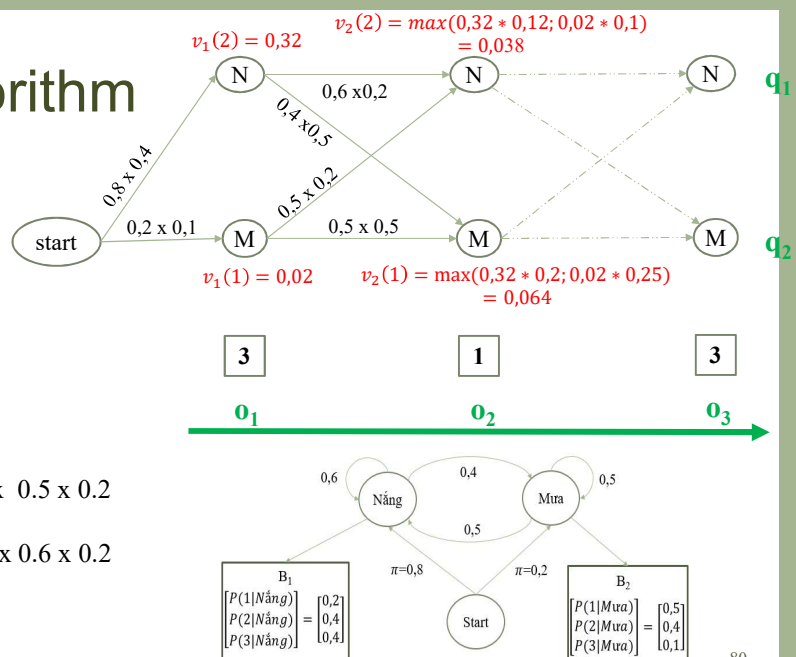
# The Viterbi algorithm

The value of each cell  $v_t(j)$  is computed by recursively taking the most probable path that could lead us to this cell

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

$$v_2(2) = v_1(1) \times P(H|C) \times P(1|H) = 0.02 \times 0.5 \times 0.2$$

$$v_2(2) = v_1(2) \times P(H|H) \times P(1|H) = 0.32 \times 0.6 \times 0.2$$



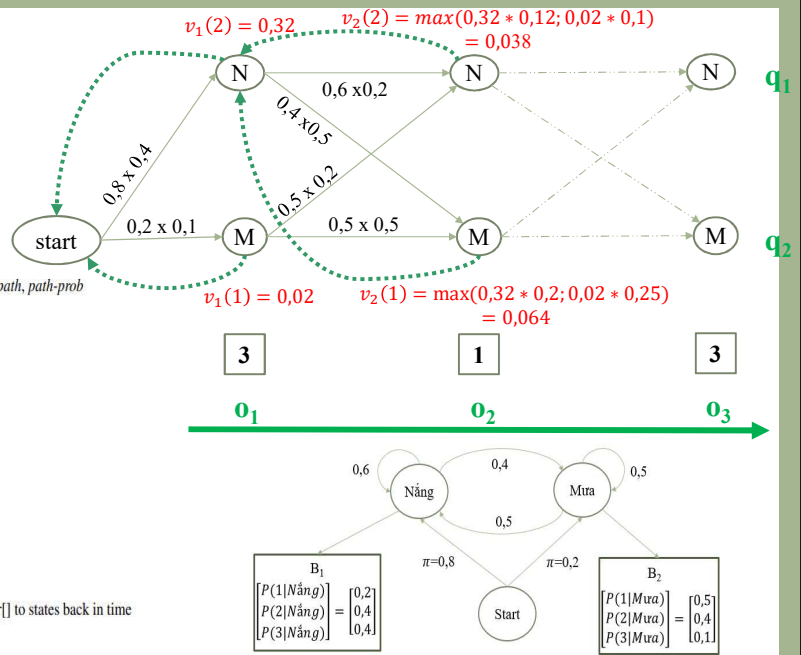
80



# The Viterbi algorithm

```

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



## The Viterbi algorithm vs. the forward algorithm

- The Viterbi algorithm is identical to the forward algorithm EXCEPT it takes the **max** over the previous path probabilities whereas the forward algorithm takes the **sum**.
- The Viterbi algorithm has one component that the forward algorithm doesn't have: **backpointers**. Why?
  - The forward algorithm needs to produce an observation likelihood,
  - The Viterbi algorithm must produce a probability and also the most likely state sequence.
    - Computing this best state sequence by keeping track of the path of hidden states that led to each state, and then at the end backtracing the best path to the beginning (the Viterbi **backtrace**).

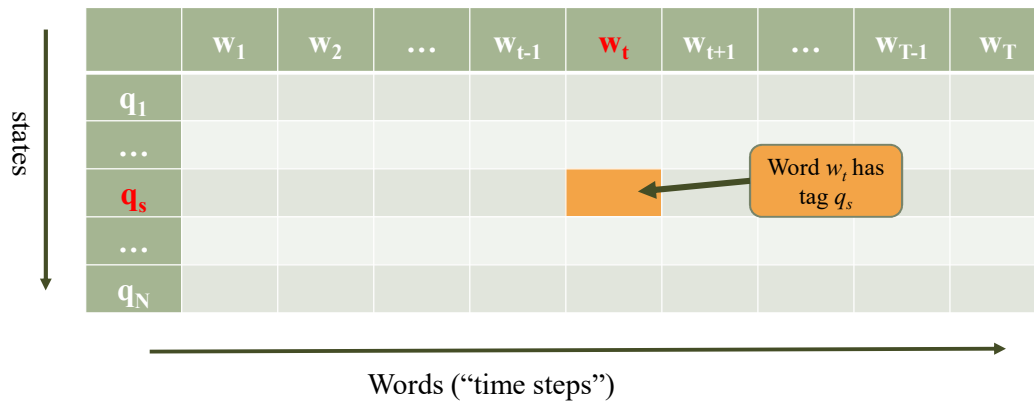
# The trellis

- We use a trellis ( $N$  rows x  $T$  columns) to keep track of the HMM.
  - State-graph of len  $N$  (number of tags is  $N$ )
  - Observation of len  $T$  (number of words in the sequence is  $T$ ),

- The HMM can assign one of the  $T$  tags to each of the  $N$  words.

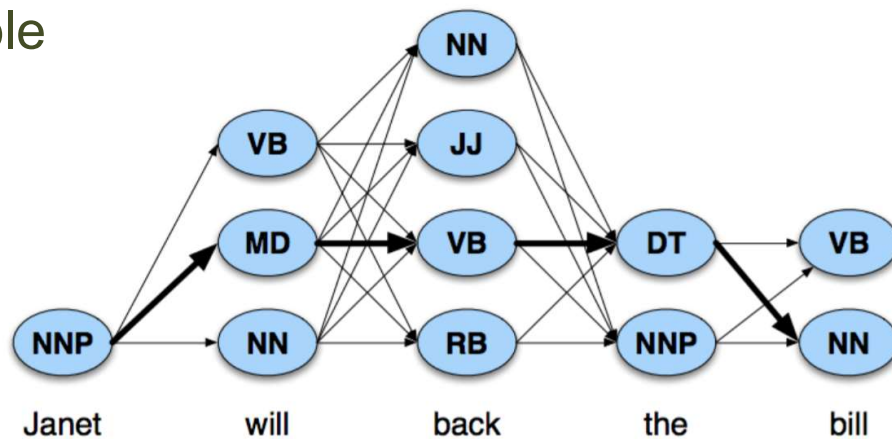
Computing  $P(t|w)$  for one tag sequence

- One path through the trellis = one tag sequence
- We just multiply the probabilities as before



LNK\_83

# Example



<https://web.stanford.edu/~jurafsky/slp3/8.pdf>

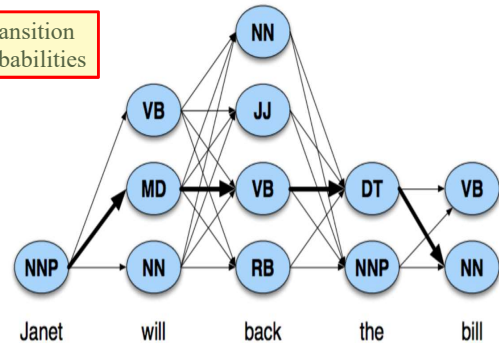
## Example – “Janet will back the bill.”

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

**Figure 8.12** The  $A$  transition probabilities  $P(t_i|t_{i-1})$  computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus  $P(VB|MD)$  is 0.7968.

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

**Figure 8.13** Observation likelihoods  $B$  computed from the WSJ corpus without smoothing, simplified slightly.



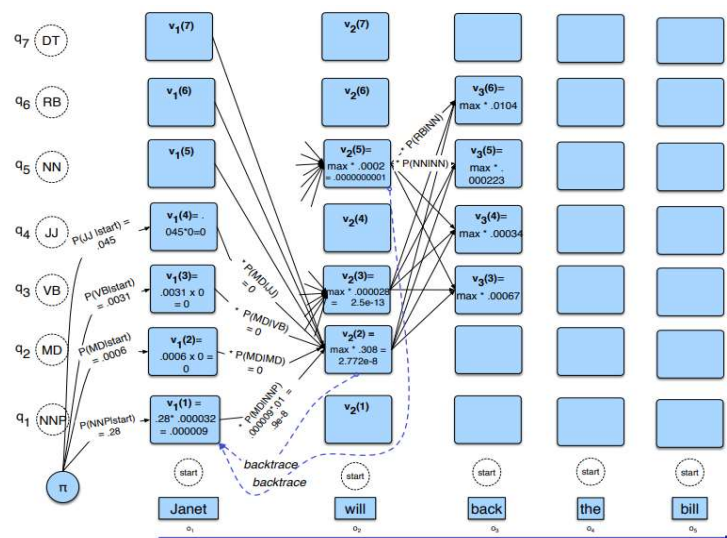
1. Use the forward algorithm
2. Use the Viterbi algorithm

85

## Example – “Janet will back the bill.”

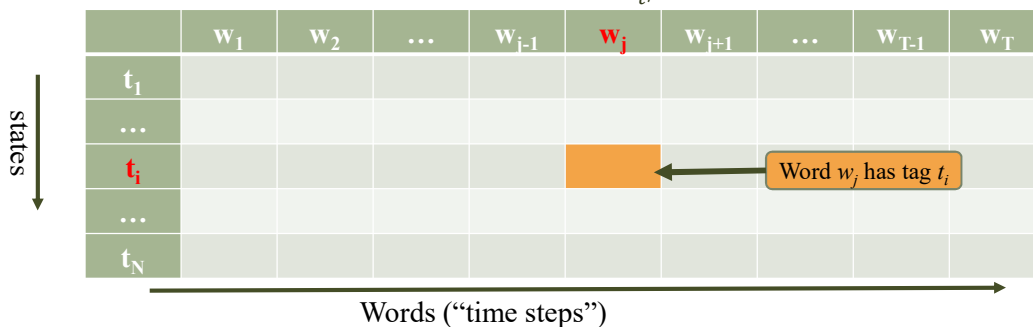
- **Figure 8.14:** The first few entries in the individual state columns for the Viterbi algorithm. Each cell keeps the probability of the best path so far and a pointer to the previous cell along that path. We have only filled out columns 1 and 2; to avoid clutter most cells with value 0 are left empty. The rest is left as an exercise for the reader. After the cells are filled in, backtracing from the end state, we should be able to reconstruct the correct state sequence

NNP MD VB DT NN



## Example – “Janet will back the bill.”

- Create a trellis ( $N$  rows x  $T$  columns)
  - State-graph of len  $N$  (number of tags is  $N$ )
  - Observation of len  $T$  (number of words in the sequence is  $T$ ),
- In the first column, for each tag in the tagset compute  $trellis[t][1] = P(t|start)P(w_1|t)$
- For each column  $j=2$  to  $T$  in the trellis
  - For each tag  $t$  in the tagset compute  $trellis[t][j] = \max_{t'} trellis[t', j-1]P(t|t')P(w_j|t)$



LNK\_87

	NNP
<s>	0.2767
NNP	0.3777
MD	0.0008
VB	0.0322
JJ	0.0366
NN	0.0096
RB	0.0068
DT	0.1147

	Janet
NNP	0.000032
MD	0
VB	0
JJ	0
NN	0
RB	0
DT	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009				
MD	0				
VB	0				
JJ	0				
NN	0				
RB	0				
DT	0				

	Janet	will
<b>NNP</b>	0.000032	0
<b>MD</b>	0	0.308431
<b>VB</b>	0	0.000028
<b>JJ</b>	0	0
<b>NN</b>	0	0.000200
<b>RB</b>	0	0
<b>DT</b>	0	0

	<b>NNP</b>	<b>MD</b>	<b>VB</b>	<b>JJ</b>	<b>NN</b>	<b>RB</b>	<b>DT</b>
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
<b>NNP</b>	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0			
MD	0	0.000009*0.011* 0.308431=3E-8			
VB	0	0.000009*0.0009* 0.000028=2.3E-13			
JJ	0	0			
NN	0	0.000009*0.0584* 0.0002=1.1E-10			
RB	0	0			
DT	0	0			

	Janet	will	back	the	bill
<b>NNP</b>	0.000032	0	0	0.000048	0
<b>MD</b>	0	0.308431	0	0	0
<b>VB</b>	0	0.000028	0.000672	0	0.000028
<b>JJ</b>	0	0	0.000340	0.000097	0
<b>NN</b>	0	0.000200	0.000223	0.000006	0.002337
<b>RB</b>	0	0	0.010446	0	0
<b>DT</b>	0	0	0	0.506099	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0		
MD	0	0.000009*0.011* 0.308431=3E-8	0		
VB	0	0.000009*0.0009* 0.000028=2.3E-13			
JJ	0	0			
NN	0	0.000009*0.0584* 0.0002=1.1E-10			
RB	0	0			
DT	0	0	0		

back

	VB	JJ	NN	RB
MD 3E-8	<b>*0.7968=2.4E-8</b>	<b>*0.0005=1.5E-11</b>	*0.0008=2.4E-11	<b>*0.009=2.7E-10</b>
VB 2.3E-13	*0.005=1.5E-15	*0.0837=1.9E-14	*0.0615=1.4E-14	*0.0514=1.2E-14
NN 1.1E-10	*0.0014=1.5E-13	*0.0086=9.5E-13	<b>*0.1216=1.3E-11</b>	*0.0177=1.9E-12

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
<b>NNP</b>	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
<b>MD</b>	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
<b>VB</b>	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
<b>JJ</b>	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
<b>NN</b>	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
<b>RB</b>	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
<b>DT</b>	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
<b>NNP</b>	0.000032	0	0	0.000048	0
<b>MD</b>	0	0.308431	0	0	0
<b>VB</b>	0	0.000028	0.000672	0	0.000028
<b>JJ</b>	0	0	0.000340	0.000097	0
<b>NN</b>	0	0.000200	0.000223	0.000006	0.002337
<b>RB</b>	0	0	0.010446	0	0
<b>DT</b>	0	0	0	0.506099	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0		
MD	0	0.000009*0.011* 0.308431=3E-8	0		
VB	0	0.000009*0.0009* 0.000028=2.3E-13	3E-8*0.7968* 0.000672=1.6E-11		
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15		
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15		
RB	0	0	3E-8*0.009* 0.010446=2.8E-12		



	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0		
MD	0	0.000009*0.011* 0.308431=3E-8	0	0	
VB	0	0.000009*0.0009* 0.000028=2.3E-13	3E-8*0.7968* 0.000672=1.6E-11	0	
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15		
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15		
RB	0	0	3E-8*0.009* 0.010446=2.8E-12	0	

93

the

	NNP	JJ	NN	DT
VB 1.6E-11	<b>*0.0322=5.2E-13</b>	<b>*0.0837=1.3E-12</b>	<b>*0.0615=9.8E-13</b>	<b>*0.2231=3.6E-12</b>
JJ 5.1E-15	*0.0366=1.9E-16	*0.0733=3.7E-16	*0.4509=2.3E-15	*0.0036=1.8E-17
NN 3E-15	*0.0096=2.9E-17	*0.0086=2.6E-17	*0.1216=3.6E-16	*0.0068=2E-17
RB 2.8E-12	*0.0068=1.9E-14	*0.1012=2.8E-13	*0.0120=3.4E-14	*0.0479=1.3E-13

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

94

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0	1.6E-11*0.0322* 0.000048=2.5E-17	
MD	0	0.000009*0.011* 0.308431=3E-8	0	0	
VB	0	0.000009*0.0009* 0.000028=2.3E-13	3E-8*0.7968* 0.000672=1.6E-11	0	
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15	1.6E-11*0.0837* 0.000097=1.3E-16	
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15	1.6E-11*0.0615* 0.000006=5.9E-18	
RB	0	0	3E-8*0.009* 0.010446=2.8E-12	0	

95

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0	1.6E-11*0.0322* 0.000048=2.5E-17	0
MD	0	0.000009*0.011* 0.308431=3E-8	0	0	0
VB	0	0.000009*0.0009* 0.000028=2.3E-13	3E-8*0.7968* 0.000672=1.6E-11	0	
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15	1.6E-11*0.0837* 0.000097=1.3E-16	0
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15	1.6E-11*0.0615* 0.000006=5.9E-18	
RB	0	0	3E-8*0.009* 0.010446=2.8E-12	0	0

96



bill

	VB	NN
NNP 2.5E-17	*0.0009=2.2E-20	*0.0584=1.5E-18
JJ 1.3E-16	*0.0001=1.3E-20	*0.4509=5.9E-17
NN 5.9E-18	*0.0014=8.3E-21	*0.1216=7.2E-19
DT 1.8E-12	<b>*0.0002=3.6E-16</b>	<b>*0.4744=8.5E-13</b>

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

97

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0	1.6E-11*0.0322* 0.000048=2.5E-17	0
MD	0	0.000009*0.011* 0.308431=3E-8	0	0	0
VB	0	0.000009*0.0009* 0.000028=2.3E-13	3E-8*0.7968* 0.000672=1.6E-11	0	1.8E-12*0.0002* 0.000028=1E-20
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15	1.6E-11*0.0837* 0.000097=1.3E-16	0
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15	1.6E-11*0.0615* 0.000006=5.9E-18	1.8E-12*0.4744* 0.002337=2E-15
RB	0	0	3E-8*0.009* 0.010446=2.8E-12	0	0

98

	Janet	will	back	the	bill
NNP	0.2767* 0.000032= 0.000009	0	0	1.6E-11*0.0322* 0.000048=2.5E-17	0
MD	0	0.000009*0.011* 0.308431=3E-8	0	0	0
VB	0	0.000009*0.0009* 0.000028=2.3E-13	3E-8*0.7968* 0.000672=1.6E-11	0	1.8E-12*0.0002* 0.000028=1E-20
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15	1.6E-11*0.0837* 0.000097=1.3E-16	0
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15	1.6E-11*0.0615* 0.000006=5.9E-18	<b>1.8E-12*0.4744* 0.002337=2E-15</b>
RB	0	0	3E-8*0.009* 0.010446=2.8E-12	0	0
DT	0	0	0	1.6E-11*0.2231* 0.506099=1.8E-12	0

99

	NNP	MD	VB	DT	NN
	Janet	will	back	the	bill
NNP	<b>0.2767*</b> <b>0.000032=</b> <b>0.000009</b>	0	0	1.6E-11*0.0322* 0.000048=2.5E-17	0
MD	0	<b>0.000009*0.011*</b> <b>0.308431=3E-8</b>	0	0	0
VB	0	0.000009*0.0009* 0.000028=2.3E-13	<b>3E-8*0.7968*</b> <b>0.000672=1.6E-11</b>	0	1.8E-12*0.0002* 0.000028=1E-20
JJ	0	0	3E-8*0.0005* 0.00034=5.1E-15	1.6E-11*0.0837* 0.000097=1.3E-16	0
NN	0	0.000009*0.0584* 0.0002=1.1E-10	1.1E-10*0.1216* 0.000223=3E-15	1.6E-11*0.0615* 0.000006=5.9E-18	<b>1.8E-12*0.4744*</b> <b>0.002337=2E-15</b>
RB	0	0	3E-8*0.009* 0.010446=2.8E-12	0	0
DT	0	0	0	<b>1.6E-11*0.2231*</b> <b>0.506099=1.8E-12</b>	0

100

## Exercise

## DATA

Nếu cần học Lan

1	Lan	Trúc	cần	hỏi	Nếu
	N	N	M	V	N
2	Học	nếu	hỏi	Lan	
	N	M	V	N	
3	Nếu	Trúc	học	Lan	
	M	N	V	N	
4	Lan	nếu	tìm	Học	
	N	M	V	N	

Solution ?

LNK\_101





## POS tagging approaches

- Rule-based POS tagging
- Statistical POS tagging
- Markov chain
- **Hidden Markov Model**
  - The forward algorithm
  - The Viterbi algorithm
  - **The forward-backward algorithm**
- HMM POS tagging

LNK\_107

## HMM – Problem 3

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

- **Learning:** Given an observation sequence  $O$  and the set of possible states in the HMM, learn the HMM parameters  $A$  and  $B$ .
- The input to such a learning algorithm would be
  - an unlabeled sequence of observations  $O$  and
  - a vocabulary of potential hidden states  $Q$ .
- The standard algorithm for HMM training is the forward-backward, or Baum-Welch algorithm (Baum, 1972), a special case of the Expectation-Maximization or EM algorithm (Dempster et al., 1977).
- The algorithm will let us **train both** the **transition probabilities  $A$**  and the **emission probabilities  $B$**  of the HMM.
- EM is an iterative algorithm, computing an initial estimate for the probabilities, then using those estimates to computing a better estimate, and so on, iteratively improving the probabilities that it learns
- *The real problem: we don't know the counts of being in any of the hidden states*
- **Solution:** The Baum-Welch algorithm solves this by **iteratively estimating the counts**. We will start with an estimate for the transition and observation probabilities and then use these estimated probabilities to derive better and better probabilities
  - Computing the *forward probability* for an observation and
  - Then, *dividing* that probability mass *among all the different paths* that contributed to this forward probability.

For the ice cream task, we would start with :

- a sequence of observations  $O = \{1, 3, 2, \dots\}$  and
- the set of hidden states  $H$  and  $C$ .

## HMM- Problem 3

The **backward probability**  $\beta$  is the probability of seeing the observations from time  $t+1$  to the end, given that we are in **state**  $i$  at **time**  $t$  (and given the automaton  $\lambda$ ):

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \dots o_T | q_t = i, \lambda)$$

The computation of  $\beta_t(i)$  by summing all the successive values  $\beta_{t+1}(j)$  weighted by their transition probabilities  $a_{ij}$  and their observation probabilities  $b_j(o_{t+1})$ . Start and end states not shown.

### 1. Initialization:

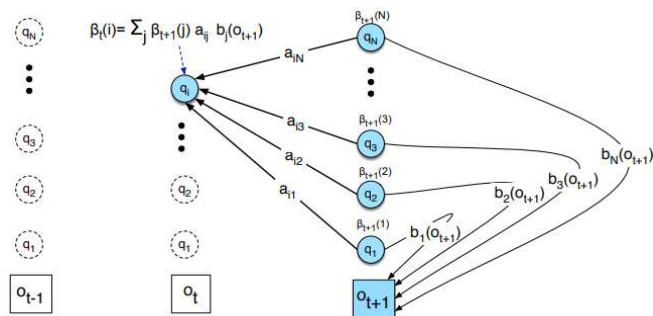
$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

### 2. Recursion

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T$$

### 3. Termination:

$$P(O|\lambda) = \sum_{j=1}^N \pi_j b_j(o_1) \beta_1(j)$$



## HMM- Problem 3

- Put all together to see how the **forward and backward probabilities** can help **compute** the **transition probability**  $a_{ij}$  and **observation probability**  $b_i(o_i)$  from an observation sequence, even though the actual path taken through the model is hidden
- Estimate  $\hat{a}_{ij}$  by a variant of simple maximum likelihood estimation

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

- Denote the  $\xi_t$  as the **probability of being in state**  $i$  at **time**  $t$  and **state**  $j$  at **time**  $t+1$ , given the observation sequence and of course the model

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | O, \lambda)$$

- To compute  $\xi_t$ , we first compute a probability which is similar to  $\xi_t$ , but differs in including the probability of the observation; note the different conditioning of  $O$

$$\text{not-quite-}\xi_t(i, j) = P(q_t = i, q_{t+1} = j, O | \lambda) \quad \text{not-quite-}\xi_t(i, j) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

## HMM- Problem 3

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\begin{aligned} \xi_t(i, j) &= P(q_t = i, q_{t+1} = j | O, \lambda) \\ \text{not-quite-}\xi_t(i, j) &= P(q_t = i, q_{t+1} = j, O | \lambda) \\ \text{not-quite-}\xi_t(i, j) &= \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \end{aligned}$$

$$P(X|Y, Z) = \frac{P(X, Y|Z)}{P(Y|Z)}$$

$$P(O|\lambda) = \sum_{j=1}^N \alpha_t(j) \beta_t(j)$$

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^N \alpha_t(j) \beta_{t+1}(j)}$$

The **expected number of transitions** from state ***i*** to state ***j*** is then the **sum over all *t*** of  **$\xi$**

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

111

## HMM- Problem 3

- We also need a formula for **recomputing the observation probability**.
  - This is the **probability of a given symbol  $v_k$  from the observation vocabulary  $V$ , given a state  $j$**

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

- Denoting  $\gamma_t(j)$  as the **probability of being in state  $j$  at time  $t$**

$$\gamma_t(j) = P(q_t = j | O, \lambda)$$

$$\gamma_t(j) = \frac{P(q_t = j, O | \lambda)}{P(O | \lambda)}$$

$$\gamma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{P(O | \lambda)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T \text{s.t. } O_t = v_k \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

112



# The forward-backward algorithm

**function** FORWARD-BACKWARD(*observations of len T, output vocabulary V, hidden state set Q*) **returns** HMM=(A,B)

**initialize** A and B

**iterate** until convergence

**E-step**

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall t \text{ and } j$$

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall t, i, \text{ and } j$$

**M-step**

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1, s.t. O_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

**return** A, B

- In the E-step, we compute the expected state occupancy count  $\gamma$  and the expected state transition count  $\xi$  from the earlier A and B probabilities.
- In the M-step, we use  $\gamma$  and  $\xi$  to recompute new A and B probabilities.

113

## Sketch of Baum-Welch (EM) Algorithm for Training HMMs

Assume an HMM with  $N$  states.

Randomly set its parameters  $\lambda=(A,B)$

(making sure they represent legal distributions)

Until converge (i.e.  $\lambda$  no longer changes) do:

- **E Step:** Use the forward/backward procedure to determine the probability of various possible state sequences for generating the training data
- **M Step:** Use these probability estimates to re-estimate values for all of the parameters  $\lambda$

114

# Self-study

- Extending the HMM algorithm to trigrams
- Beam Search
- Maximum Entropy Markov models

115

1. Introduction

2. POS tagging  
approaches

3. Evaluation

116

## Evaluation metric: test accuracy

- How many words in the unseen test data can you tag correctly?
  - State of the art on Penn Treebank: around 97%.
  - $\Rightarrow$  How many sentences can you tag correctly?
- Compare your model against a baseline
  - Standard: assign to each word its most likely tag (use training corpus to estimate  $P(t|w)$  )
  - Baseline performance on Penn Treebank: around 93.7%
- ... and a (human) ceiling
  - How often do human annotators agree on the same tag? Penn Treebank: around 97%

LNK\_117

## 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.
- Evaluation performance
  - How do we know how well a tagger does?
  - Say we had a test sentence, or a set of test sentences, that were already tagged by a human (a “Gold Standard”)
  - We could run a tagger on this set of test sentences
  - And see how many of the tags we got right.
  - This is called “Tag accuracy” or “Tag percent correct”

118

## Computing % correct

- Of all the words in the test set
- For what percent of them did the tag chosen by the tagger equal the human-selected tag.

$$\%correct = \frac{\text{\# of words tagged correctly in test set}}{\text{total \# of words in test set}}$$

- Human tag set: (“Gold Standard” set)

## Training and Test sets

- Often they come from the same labeled corpus!
- We just use 90% of the corpus for training and save out 10% for testing!
- Even better: cross-validation
  - Take 90% training, 10% test, get a % correct
  - Now take a different 10% test, 90% training, get % correct
  - Do this 10 times and average

## Evaluation and rule-based taggers

- Does the same evaluation metric work for rule-based taggers?
- Yes!
  - Rule-based taggers don't need the training set
  - But they still need a test set to see how well the rules are working

## Qualitative evaluation

- Generate a **confusion matrix** (for development data): How often was a word with tag  $i$  mistagged as tag  $j$ :

		Correct Tags						
		IN	JJ	NN	NNP	RB	VBD	VBN
Predicted Tags	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	—

% of errors caused by mistagging VBN as JJ

- See what errors are causing problems:
  - Noun (NN) vs. ProperNoun (NNP) vs. Adj (JJ)
  - Preterite (VBD) vs. Participle (VBN) vs. Adjective (JJ)

