



COS 484

Natural Language Processing

L6: Sequence Models

Spring 2023

Lecture plan

This lecture

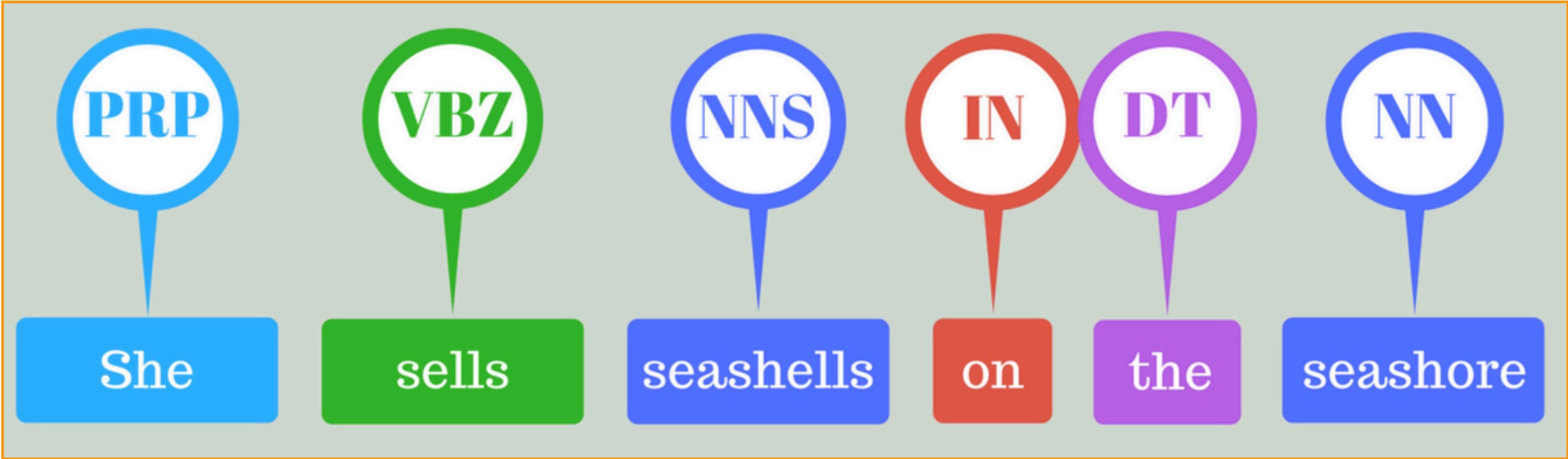
- Hidden markov models
- Viterbi algorithm

Next lecture

- Maximum entropy markov model (MEMMs)
- Conditional random field (CRFs)

We will also learn sequence tagging NLP tasks such as **part-of-speech tagging** or **named entity recognition**..

Why model sequences?



PRP: Personal pronoun

VBZ: Verb, 3rd person singular present

NN: singular noun
NNS: plural noun

IN: preposition or subordinating conjunction

DT: determiner

Part-of-speech (POS) tagging

Why model sequences?

The screenshot shows a text snippet with several entities highlighted in colored boxes. Above the text, there is a legend with colored boxes and letters: Person (p), Loc (l), Org (o), Event (e), Date (d), and Other (z). The text snippet is: "Barack Hussein Obama II (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate." The entities are: "Barack Hussein Obama II" (Person, p), "August 4, 1961" (Date, d), "American" (Other, z), "the United States" (Loc, l), "January 20, 2009" (Date, d), "January 20, 2017" (Date, d), "Democratic Party" (Org, o), "African American" (Other, z), "United States Senator" (Other, z), "Illinois" (Loc, l), and "Illinois State Senate" (Org, o).

Named Entity recognition

Why model sequences?

Mary loaded the truck with hay at the depot on Friday.

load.01

A0 loader

A1 bearer

A2 cargo

A3 instrument

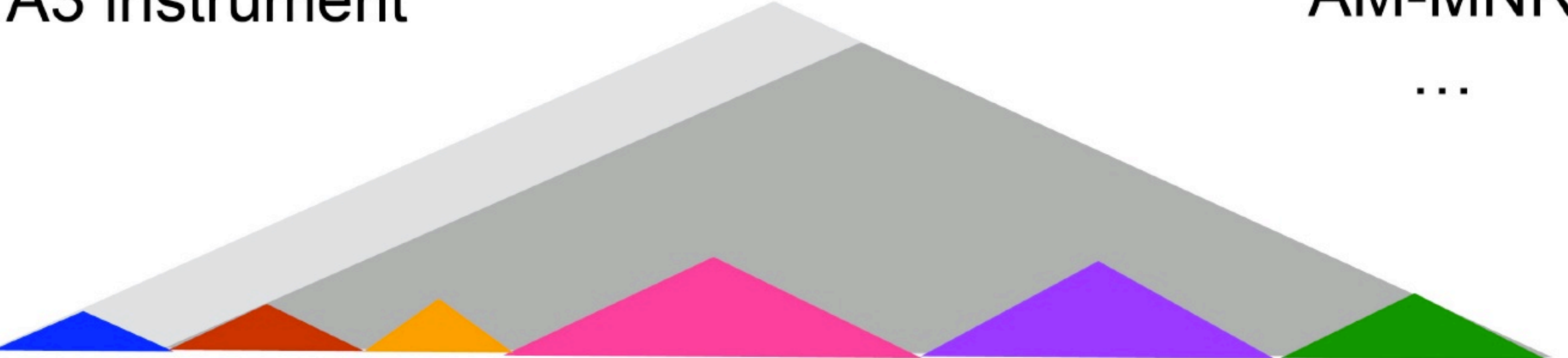
AM-LOC

AM-TMP

AM-PRP

AM-MNR

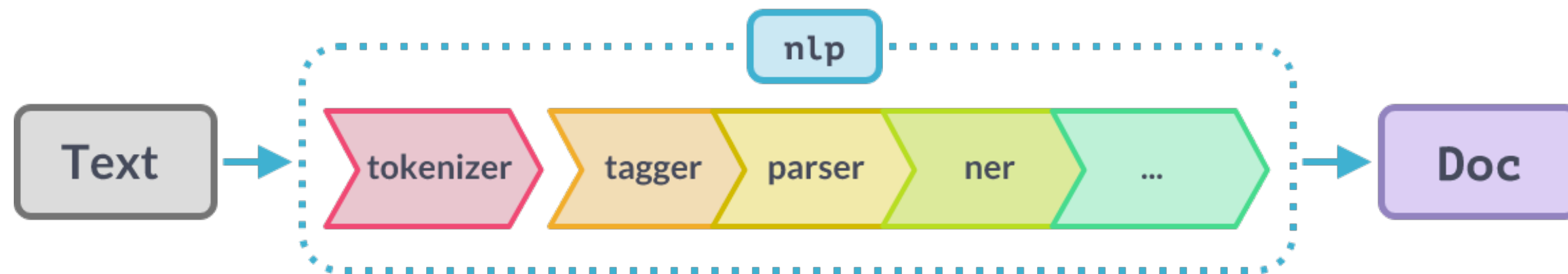
...



Mary loaded hay onto the truck at the depot on Friday.

Semantic role labeling

NLP pipelines



NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	Tokenizer	Doc	Segment text into tokens.
PROCESSING PIPELINE			
tagger	Tagger	Token.tag	Assign part-of-speech tags.
parser	DependencyParser	Token.head , Token.dep , Doc.sents , Doc.noun_chunks	Assign dependency labels.
ner	EntityRecognizer	Doc.ents , Token.ent_iob , Token.ent_type	Detect and label named entities.

<https://spacy.io/usage/processing-pipelines>

Part of speech:

NNP NNP RB VBD IN NNP NNP CC PRP VBZ RB VBG PRP IN PRP .
 Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him .

Named entity recognition:

Person Date Person Date
 Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

Co-reference:

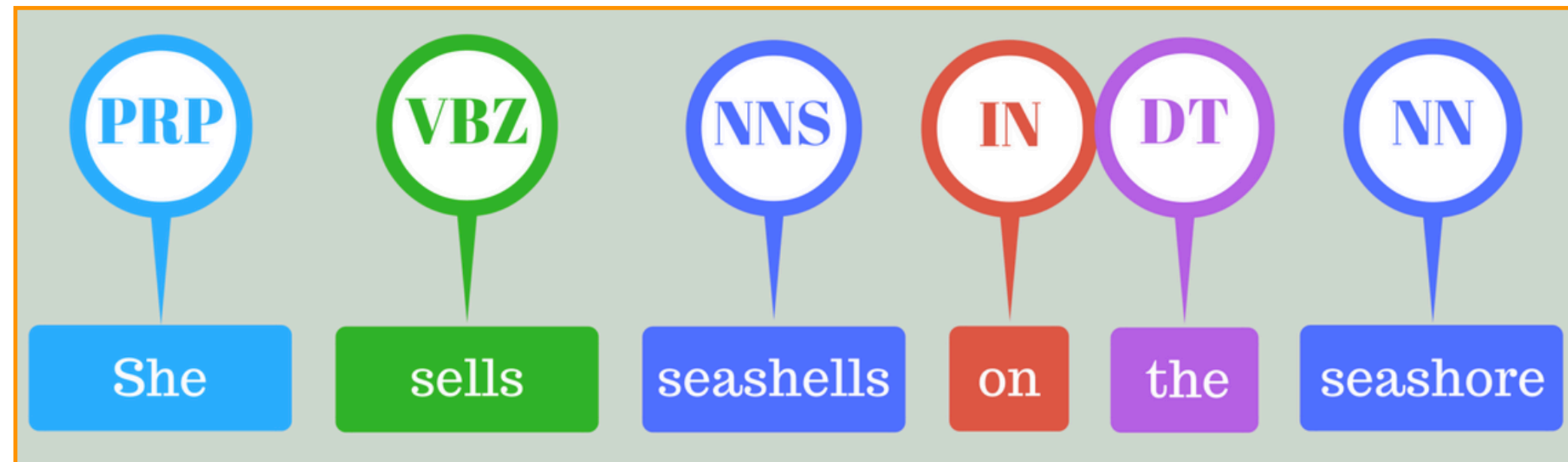
Mention Ment M Mention M
 Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

Basic dependencies:

compound nsubj nmod case compound nsubj aux advmod dobj nmod
 Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him .

<https://stanfordnlp.github.io/CoreNLP/pipeline.html>

What are part of speech tags?

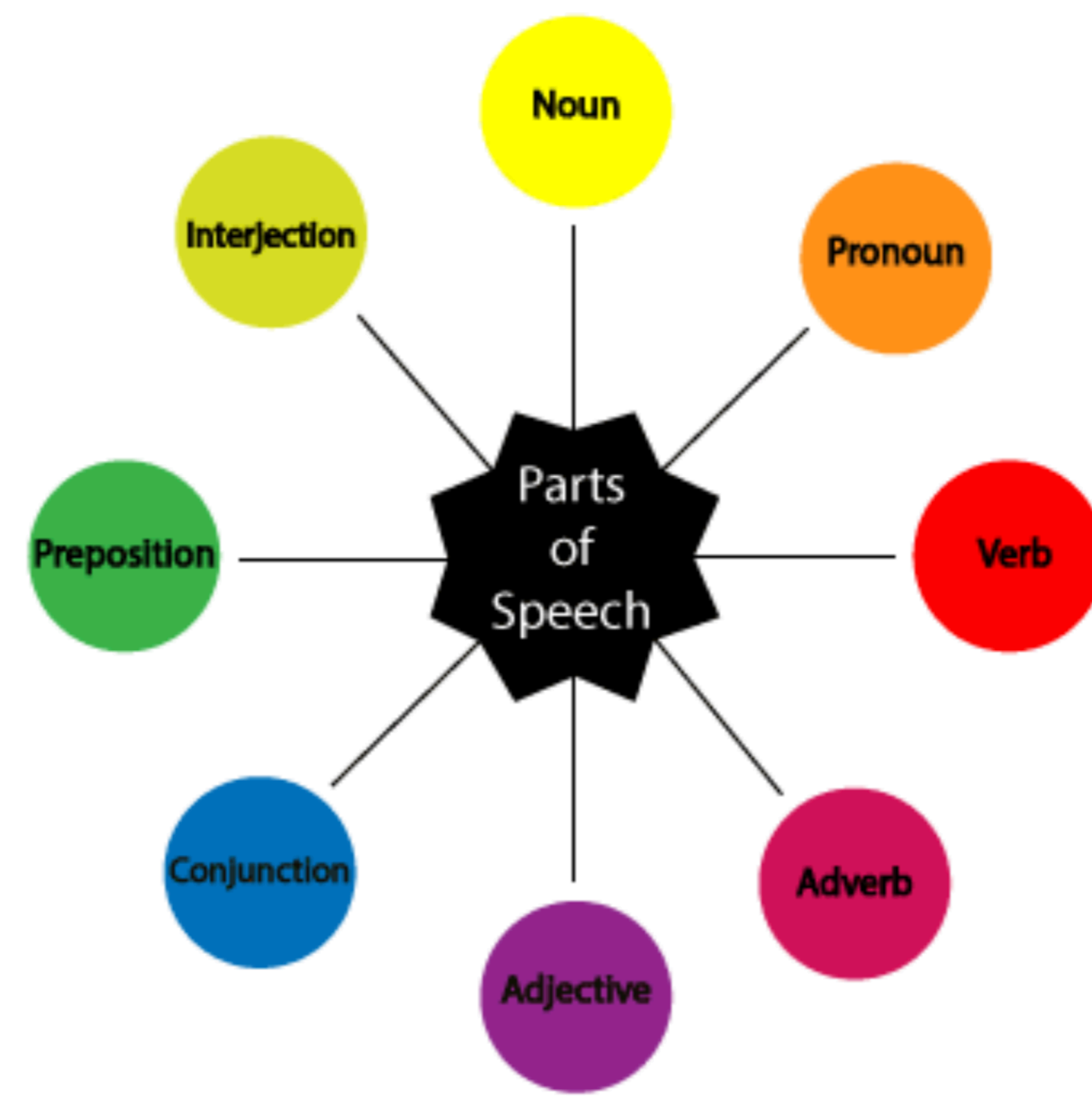


- Word classes or syntactic categories
- Reveal useful information about a word (and its neighbors!)

1. The/**DT** cat/**NN** sat/**VBD** on/**IN** the/**DT** mat/**NN**
2. Princeton/**NNP** is/**VBZ** in/**IN** New/**NNP** Jersey/**NNP**
3. The/**DT** old/**NN** man/**VBP** the/**DT** boat/**NN**

Parts of Speech

- Different words have different functions
- Can be roughly divided into two classes
- **Closed class:** fixed membership, **function words**
 - e.g. prepositions (*in, on, of*), determiners (*the, a*)
- **Open class:** New words get added frequently
 - e.g. nouns (Twitter, Facebook), verbs (google), adjectives, adverbs





Parts of Speech

How many part of speech tags do you think English has?

- A) < 10
- B) 10 - 20
- C) 20 - 40
- D) > 40

The answer is (D) - well, depends on definitions!



Penn treebank part-of-speech tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	<i>and, but, or</i>	PDT	predeterminer	<i>all, both</i>	VBP	verb non-3sg present	<i>eat</i>
CD	cardinal number	<i>one, two</i>	POS	possessive ending	<i>'s</i>	VBZ	verb 3sg pres	<i>eats</i>
DT	determiner	<i>a, the</i>	PRP	personal pronoun	<i>I, you, he</i>	WDT	wh-determ.	<i>which, that</i>
EX	existential 'there'	<i>there</i>	PRP\$	possess. pronoun	<i>your, one's</i>	WP	wh-pronoun	<i>what, who</i>
FW	foreign word	<i>mea culpa</i>	RB	adverb	<i>quickly</i>	WP\$	wh-possess.	<i>whose</i>
IN	preposition/ subordin-conj	<i>of, in, by</i>	RBR	comparative adverb	<i>faster</i>	WRB	wh-adverb	<i>how, where</i>
JJ	adjective	<i>yellow</i>	RBS	superlatv. adverb	<i>fastest</i>	\$	dollar sign	<i>\$</i>
JJR	comparative adj	<i>bigger</i>	RP	particle	<i>up, off</i>	#	pound sign	<i>#</i>
JJS	superlative adj	<i>wildest</i>	SYM	symbol	<i>+, %, &</i>	“	left quote	<i>‘ or “</i>
LS	list item marker	<i>1, 2, One</i>	TO	“to”	<i>to</i>	”	right quote	<i>’ or ”</i>
MD	modal	<i>can, should</i>	UH	interjection	<i>ah, oops</i>	(left paren	<i>[, (, {, <</i>
NN	sing or mass noun	<i>llama</i>	VB	verb base form	<i>eat</i>)	right paren	<i>],), }, ></i>
NNS	noun, plural	<i>llamas</i>	VBD	verb past tense	<i>ate</i>	,	comma	<i>,</i>
NNP	proper noun, sing.	<i>IBM</i>	VBG	verb gerund	<i>eating</i>	.	sent-end punc	<i>. ! ?</i>
NNPS	proper noun, plu.	<i>Carolinas</i>	VBN	verb past part.	<i>eaten</i>	:	sent-mid punc	<i>: ; ... - -</i>

45 tags

(Marcus et al., 1993)

based on Wall Street
Journal (WSJ)

Other corpora: Brown, Switchboard

Part of speech tagging

- Tag each word in a sentence with its part of speech
- Disambiguation task: each word might have different functions in different contexts
- The/DT **man/NN** bought/VBD a/DT boat/NN
- The/DT old/NN **man/VBP** the/DT boat/NN

Same word,
different tags

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

Some words have
many functions!

JJ: adjective, NN: single or mass noun, VBP: Verb, non-3rd person singular present
VB: Verb, base form, RP: particle, RB: adverb

Part of speech tagging

- Tag each word in a sentence with its part of speech
- Disambiguation task: each word might have different senses/functions

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%)

Unambiguous types:
Jane → NNP,
hesitantly → RB

- Types = distinct words in the corpus
- Tokens = all words in the corpus (can be repeated)

A simple baseline



- Many words might be easy to tag
- **Most frequent class:** Assign each word to the class it occurred most in the training set. (e.g. man/NN)

How accurate do you think this baseline would be at tagging words?

- A) <50%
- B) 50-75%
- C) 75-90%
- D) >90%

The answer is (D)

A simple baseline

- Many words might be easy to tag
- **Most frequent class:** Assign each word to the class it occurred most in the training set. (e.g. man/NN)
- Accurately tags **92.34%** of word tokens on Wall Street Journal (WSJ)!
- State of the art ~ 97%
- Average English sentence ~14 words
 - Sentence level accuracies: $0.92^{14} = \mathbf{31\%}$ vs $0.97^{14} = \mathbf{65\%}$
- POS tagging not solved yet!

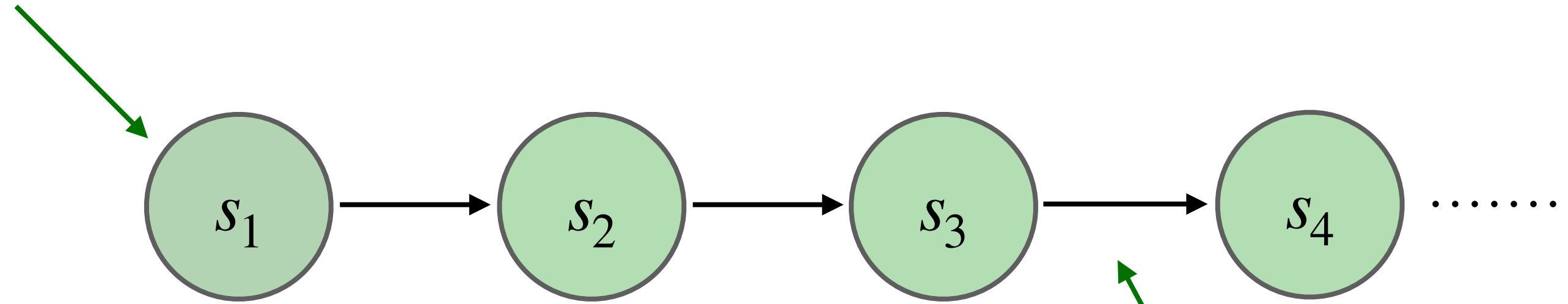
Some observations

- The function (or POS) of a word depends on its context
 - The/DT **old/JJ** **man/NN** bought/VBP the/DT boat/NN
 - The/DT **old/NN** **man/VBP** the/DT boat/NN
- Certain POS combinations are extremely unlikely
 - $\langle JJ, DT \rangle$ (“good the”) or $\langle DT, IN \rangle$ (“the in”)
- Better to make decisions on entire sentences instead of individual words

Hidden Markov Models

Markov chains

$\pi(s_1)$: initial distribution

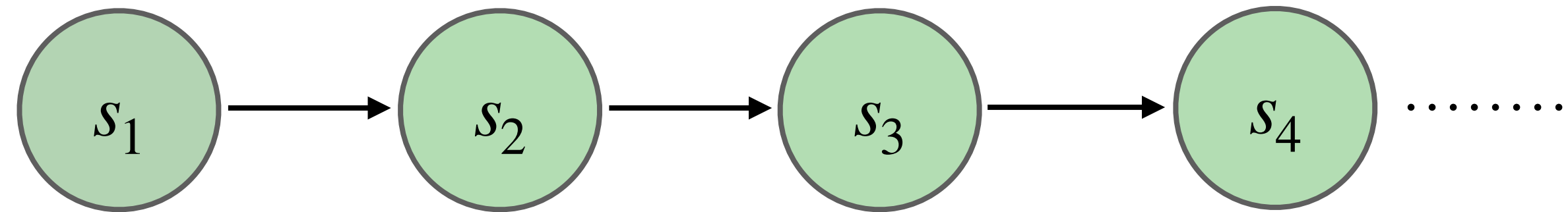


Where have we seen this before?
N-gram language models!

$p(s_t | s_{t-1})$: transition probability

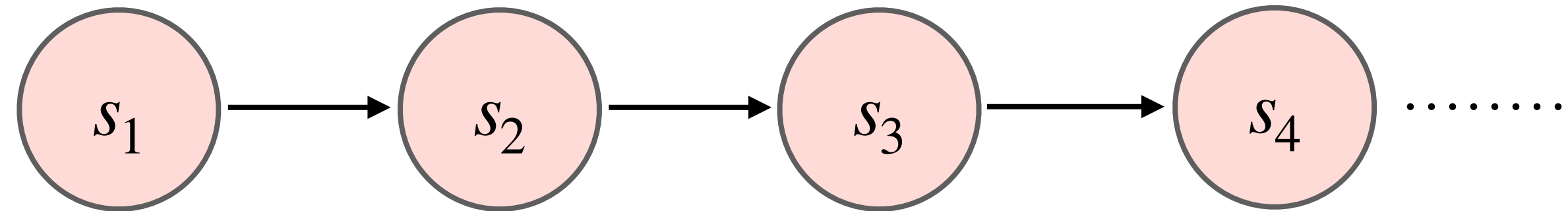
- Model probabilities of **sequences** of variables
- Each state can take one of K values (can assume $\{1, 2, \dots, K\}$ for simplicity)
- Markov assumption: $P(s_t | s_1, s_2, \dots, s_{t-1}) \approx P(s_t | s_{t-1})$
- A Markov chain is specified by
 - Initial probability distribution $\pi(s), \forall s \in \{1, \dots, K\}$
 - Transition probability matrix ($K \times K$)

Markov chains



The/**DT** cat/**NN** sat/**VBD** on/**IN** the/**DT** mat/**NN**

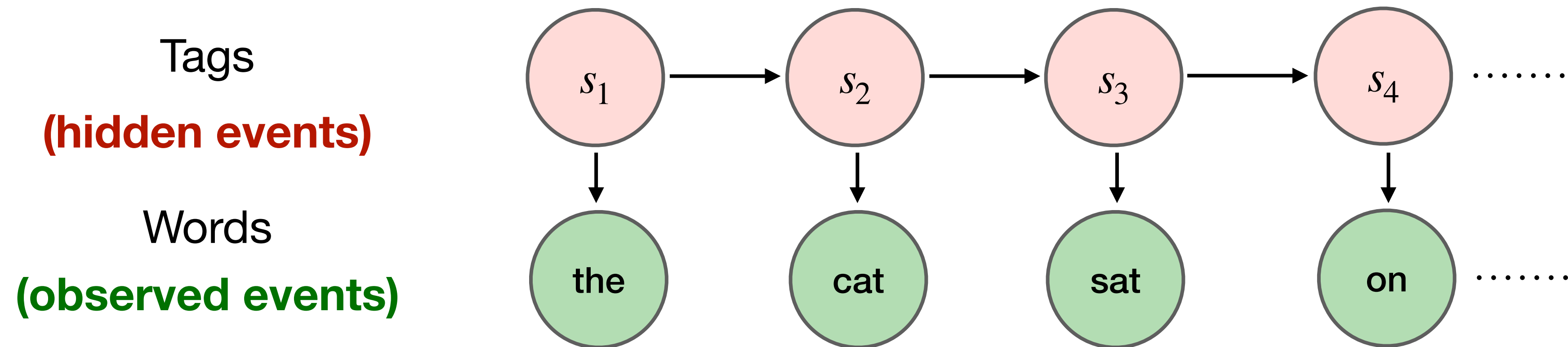
Markov chains can help us model entire sentences.



The/**??** cat/**??** sat/**??** on/**??** the/**??** mat/**??**

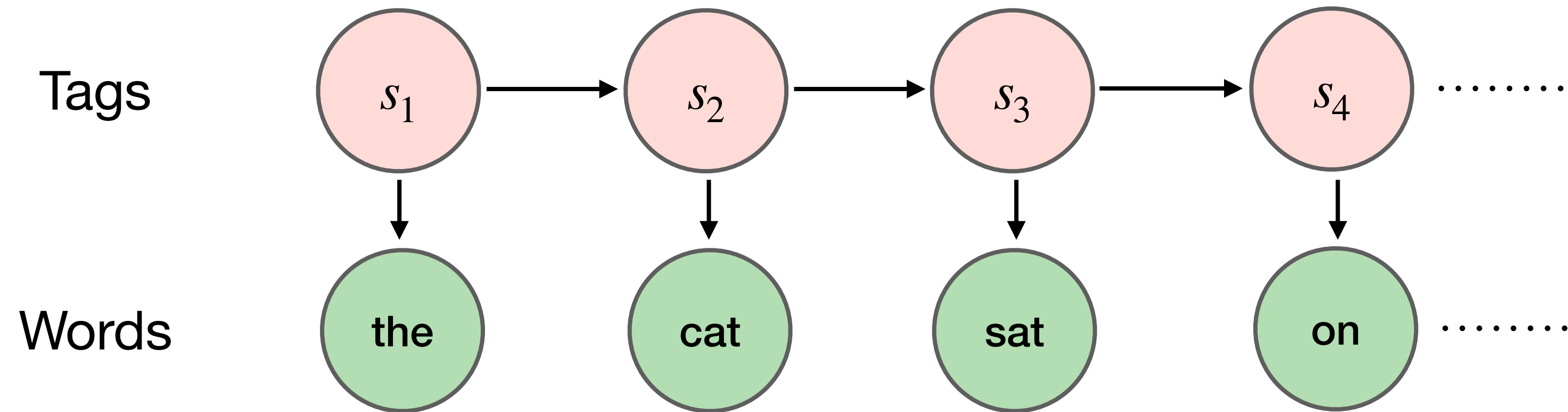
BUT we don't normally see sequences of POS tags appearing in text

Hidden Markov Model (HMM)



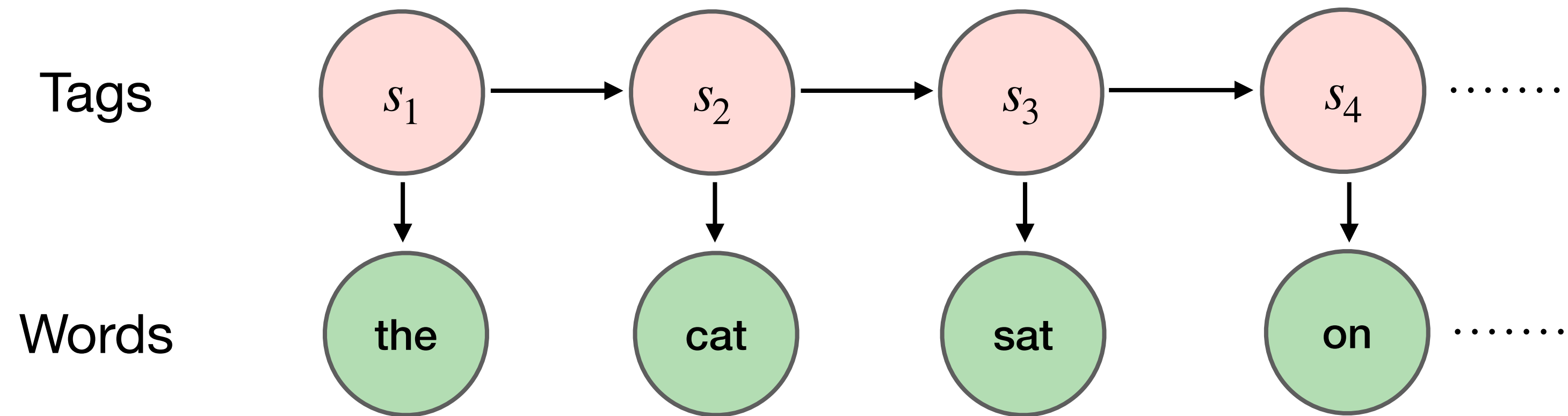
- We don't normally see sequences of POS tags in text
- However, we do observe the words!
- The HMM allows us to *jointly reason* over both **hidden** and **observed** events.
- Assume that each position has a tag that generates a word

Components of an HMM



1. Set of states $S = \{1, 2, \dots, K\}$ and set of observations $O = \{o_1, \dots, o_n\}$
2. **Initial state probability distribution** $\pi(s_1)$ $o_i \in V$
3. **Transition probabilities** $P(s_{t+1} | s_t)$
4. **Emission probabilities** $P(o_t | s_t)$

Assumptions



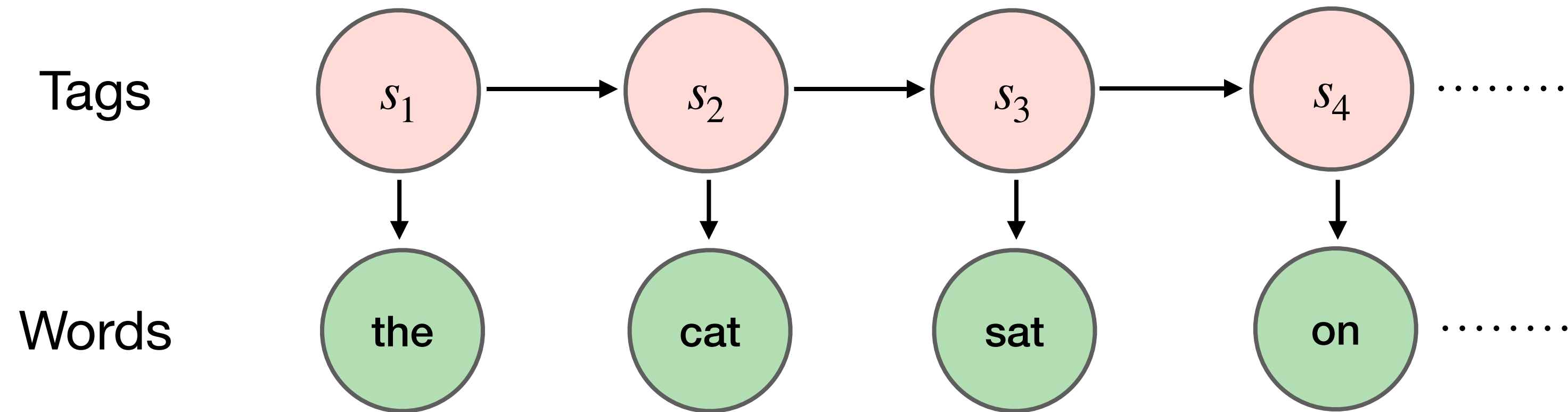
1. Markov assumption:

$$P(s_t | s_1, \dots, s_{t-1}) \approx P(s_t | s_{t-1})$$

2. Output independence:

$$P(o_t | s_1, \dots, s_t) \approx P(o_t | s_t)$$

Sequence likelihood



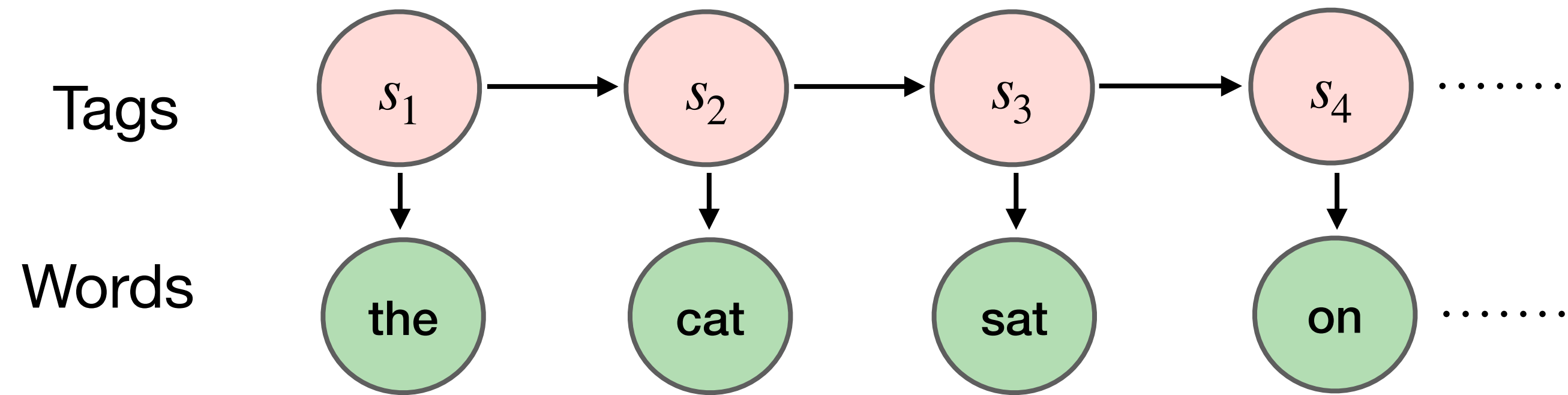
$$\begin{aligned} P(S, O) &= P(s_1, s_2, \dots, s_n, o_1, o_2, \dots, o_n) \\ &= \pi(s_1) p(o_1 | s_1) \prod_{i=2}^n P(s_i | s_{i-1}) P(o_i | s_i) \end{aligned}$$

transition probability emission probability

If we add a dummy state $s_0 = \emptyset$ at the beginning,

$$P(S, O) = \prod_{i=1}^n P(s_i | s_{i-1}) P(o_i | s_i) \quad [\pi(s_1) = P(s_1 | \emptyset)]$$

Example: Sequence likelihood



Dummy start state

	s_{t+1}	
	DT	NN
\emptyset	0.8	0.2
DT	0.2	0.8
NN	0.3	0.7

	o_t	
	the	cat
DT	0.9	0.1
NN	0.5	0.5

What is the joint probability $P(\text{the cat}, DT NN)$?

- A) $(0.8 * 0.8) * (0.9 * 0.5)$
- B) $(0.2 * 0.8) * (0.9 * 0.5)$
- C) $(0.3 * 0.7) * (0.5 * 0.5)$
- D) $(0.8 * 0.2) * (0.5 * 0.1)$

The answer is (A).

Learning

Training set:

1 Pierre/NNP Vinken/NNP ,/, 61/CD years/NNS old/JJ ,/, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.

2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP ,/, the/DT Dutch/NNP publishing/VBG group/NN ./.

3 Rudolph/NNP Agnew/NNP ,/, 55/CD years/NNS old/JJ and/CC chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP ,/, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN ./.

...

38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD people/NNS out/IN of/IN Puerto/NNP Rico/NNP ,/, who/WP were/VBD helping/VBG Hurricane/NNP Hugo/NNP victims/NNS ,/, and/CC sending/VBG them/PRP to/TO San/NNP Francisco/NNP instead/RB ./.

Maximum likelihood estimates:

$$P(s_i | s_j) = \frac{\text{Count}(s_j, s_i)}{\text{Count}(s_j)}$$

$$P(o | s) = \frac{\text{Count}(s, o)}{\text{Count}(s)}$$

Q: How many probabilities to estimate?

A: transition probabilities - $(K + 1) \times K$

emission probabilities - $|V| \times K$

Learning example

1. The/**DT** cat/**NN** sat/**VBD** on/**IN** the/**DT** mat/**NN**
2. Princeton/**NNP** is/**VBZ** in/**IN** New/**NNP** Jersey/**NNP**
3. The/**DT** old/**NN** man/**VBP** the/**DT** boat/**NN**

Maximum likelihood estimates:

$$P(s_i | s_j) = \frac{\text{Count}(s_j, s_i)}{\text{Count}(s_j)}$$

$$P(o | s) = \frac{\text{Count}(s, o)}{\text{Count}(s)}$$

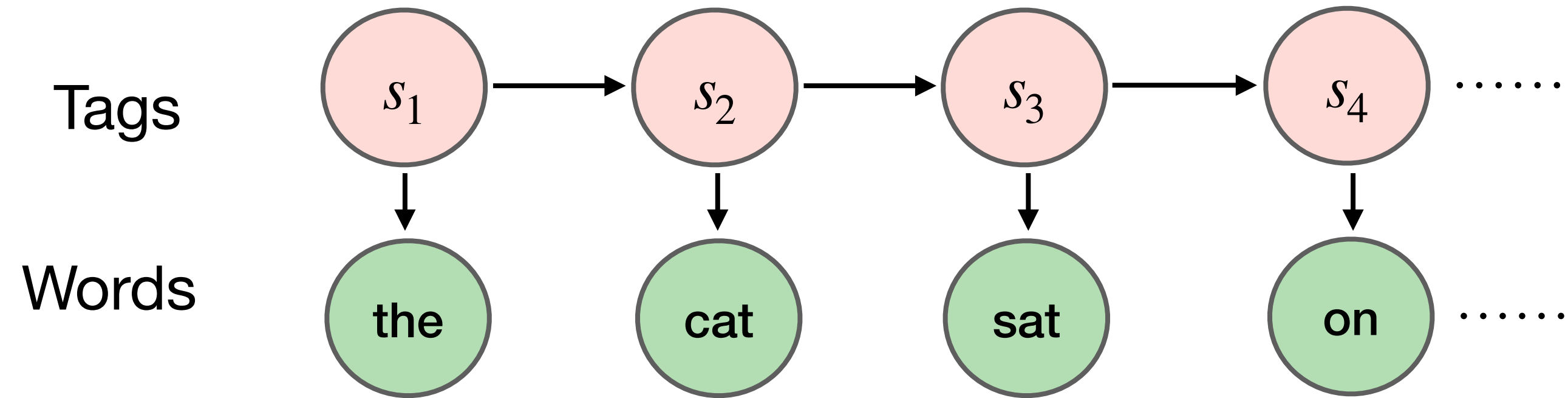
$$\pi(DT) = P(DT | \emptyset) = 2/3$$

$$P(NN | DT) = 4/4 \quad P(DT | IN) = 1/2$$

$$P(cat | NN) = 1/4 \quad P(the | DT) = 2/4$$

(assuming we
differentiate cased
vs uncased words)

Decoding with HMMs



Task: Find the most probable sequence of states $S = s_1, s_2, \dots, s_n$ given the observations $O = o_1, o_2, \dots, o_n$

$$\hat{S} = \arg \max_S P(S | O) = \arg \max_S \frac{P(O | S)P(S)}{P(O)}$$

[Bayes' Rule]

$$= \arg \max_S P(O | S)P(S)$$

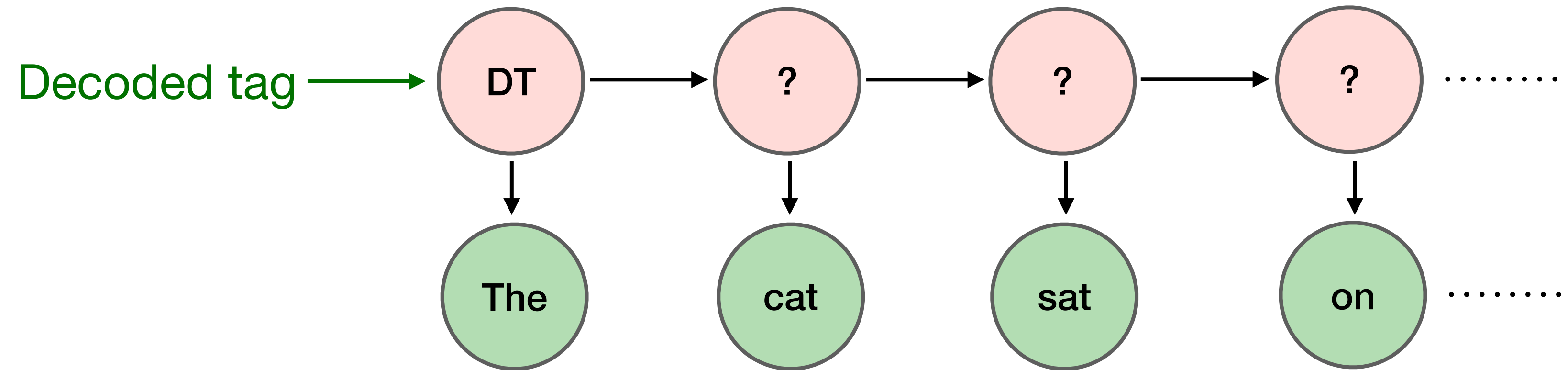
$$= \arg \max_{s_1, \dots, s_n} \prod_{i=1}^n P(s_i | s_{i-1})P(o_i | s_i)$$

How can we maximize this?
Search over all state sequences?

The Viterbi Algorithm

Greedy search

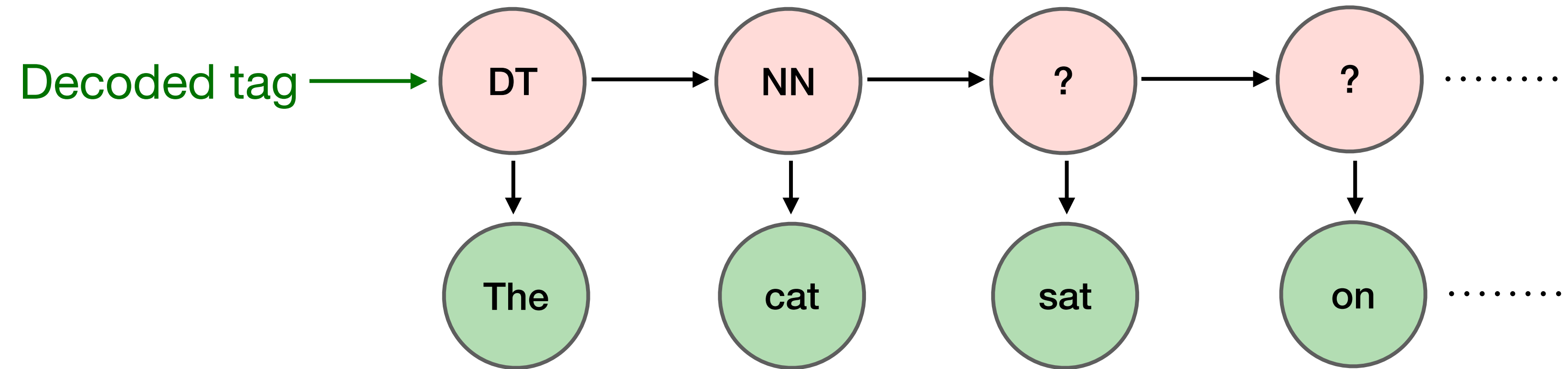
- Decode one state at a time



$$\arg \max_s \pi(s_1 = s) p(\text{The} \mid s) = \text{DT}$$

Greedy search

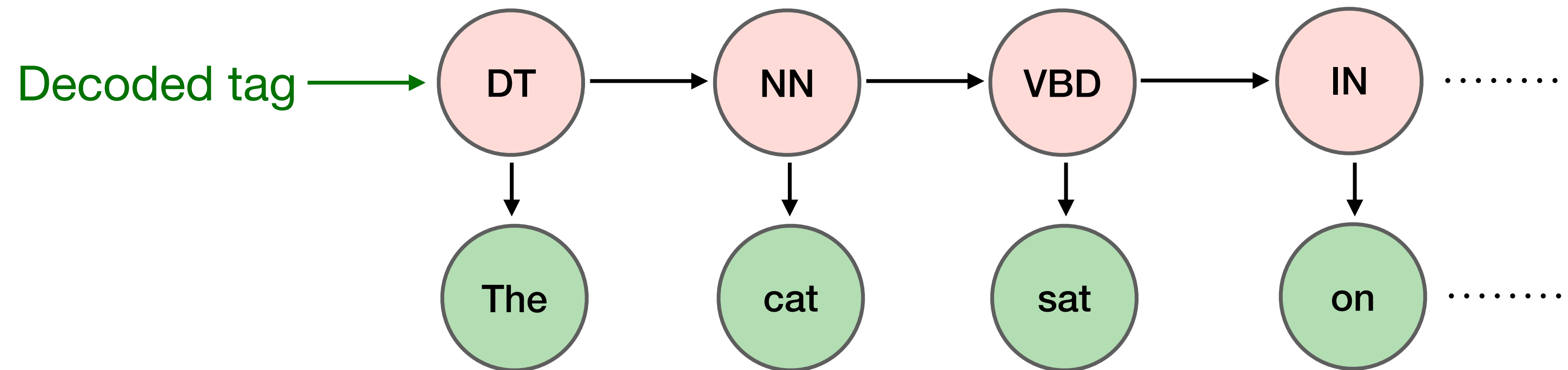
- Decode one state at a time



$$\arg \max_s p(s \mid DT)p(\text{cat} \mid s) = \text{NN}$$

Greedy search

- Decode one state at a time



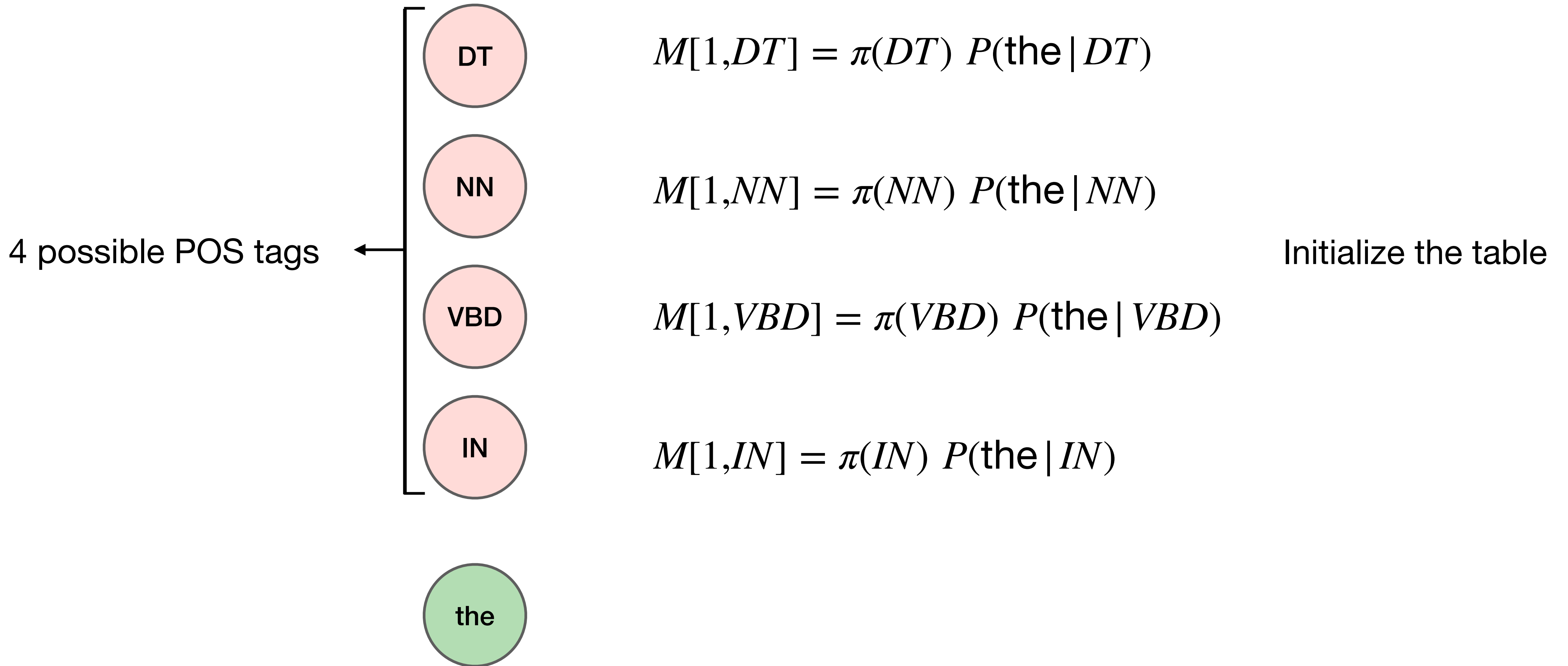
$$\hat{s}_t = \arg \max_s p(s \mid \hat{s}_{t-1})p(o_t \mid s)$$

Very efficient but it doesn't guarantee to produce the overall optimal sequence

Viterbi decoding

- Use **dynamic programming!**
- Maintain some extra data structures
- Probability lattice, $M[T, K]$ and backtracking matrix, $B[T, K]$
 - T : Number of time steps
 - K : Number of states
- $M[i, j]$ stores joint probability of most probable sequence of states ending with state j at time i , and $B[i, j]$ is the tag at time $i-1$ in the most probable sequence ending with tag j at time i

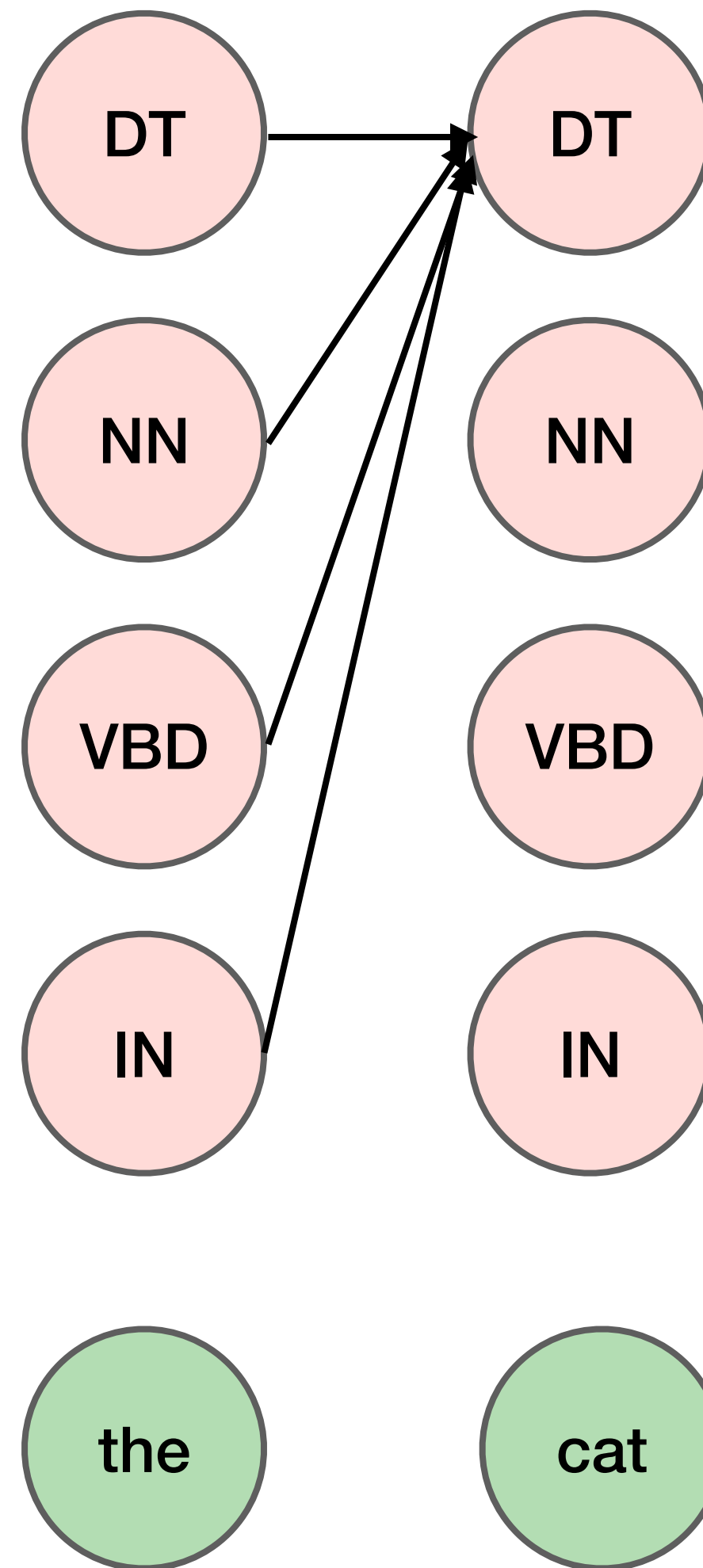
Viterbi decoding



Forward

Viterbi decoding

Consider all possible previous tags



$$M[2,DT] = \max_k M[1,k] P(DT|k) P(\text{cat}|DT)$$

$$M[2,NN] = \max_k M[1,k] P(NN|k) P(\text{cat}|NN)$$

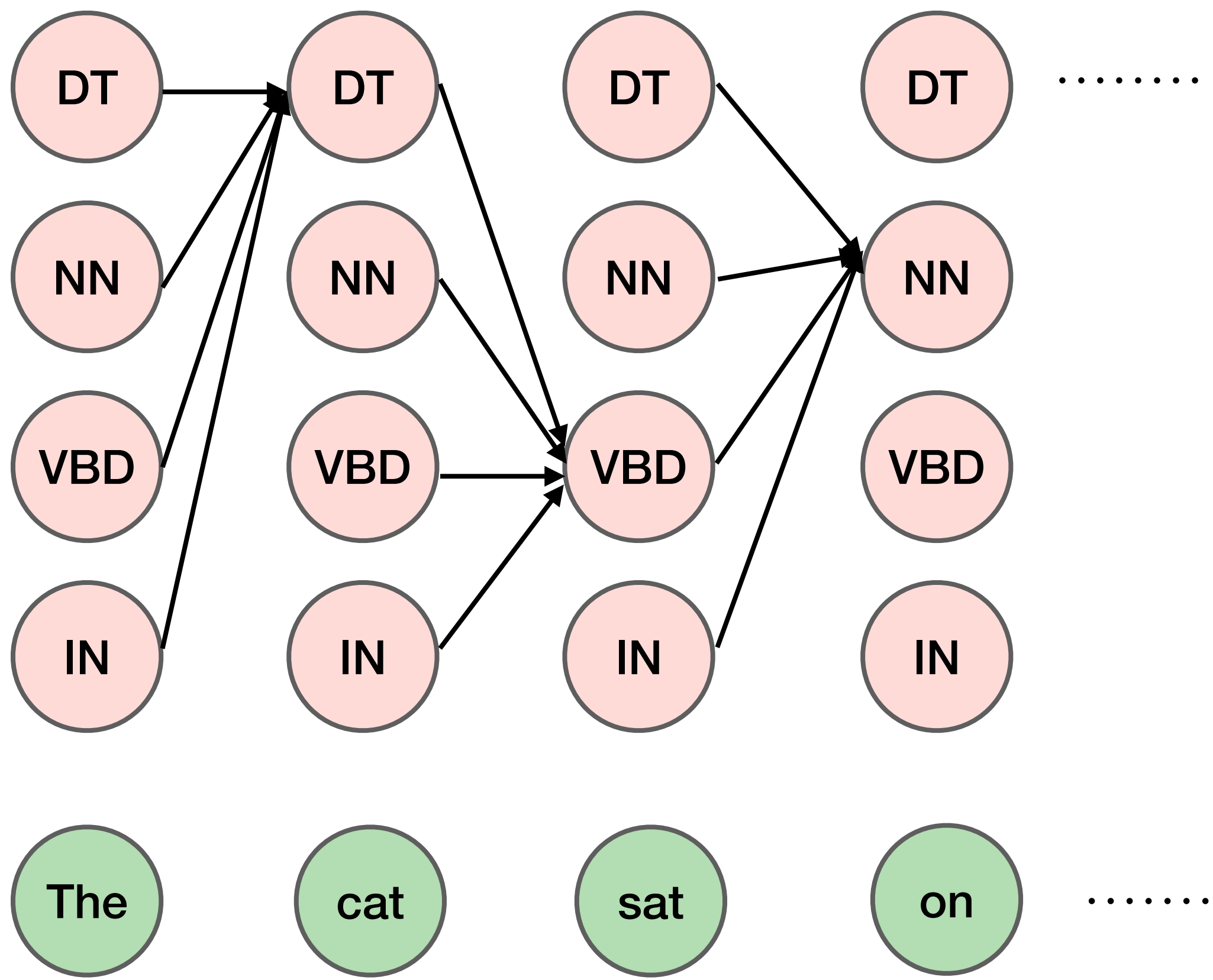
$$M[2,VBD] = \max_k M[1,k] P(VBD|k) P(\text{cat}|VBD)$$

$$M[2,IN] = \max_k M[1,k] P(IN|k) P(\text{cat}|IN)$$

Forward



Viterbi decoding



What is the time complexity of this algorithm?

- A) $O(n)$
- B) $O(nK)$
- C) $O(nK^2)$
- D) $O(n^2K)$

The answer is (C).

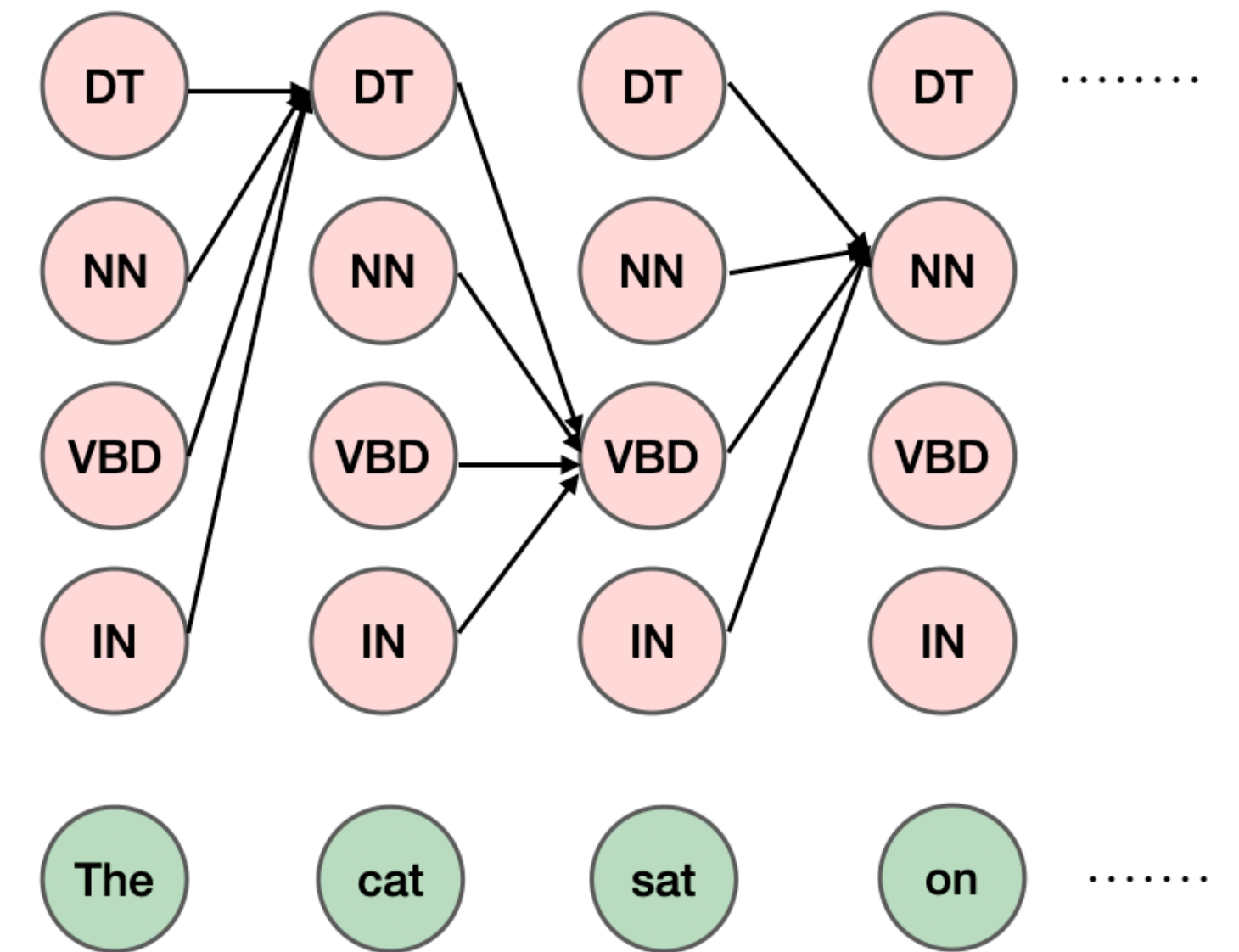
n = number of timesteps
K = number of states

$$M[i, j] = \max_k M[i - 1, k] P(s_j | s_k) P(o_i | s_j) \quad 1 \leq k \leq K \quad 1 \leq i \leq n$$

Viterbi decoding

Backward: Pick $\max_k M[n, k]$ and backtrack using B

- In practice, we maximize sum of log probabilities (or minimize the sum of negative log probabilities) instead of maximize the product of probabilities

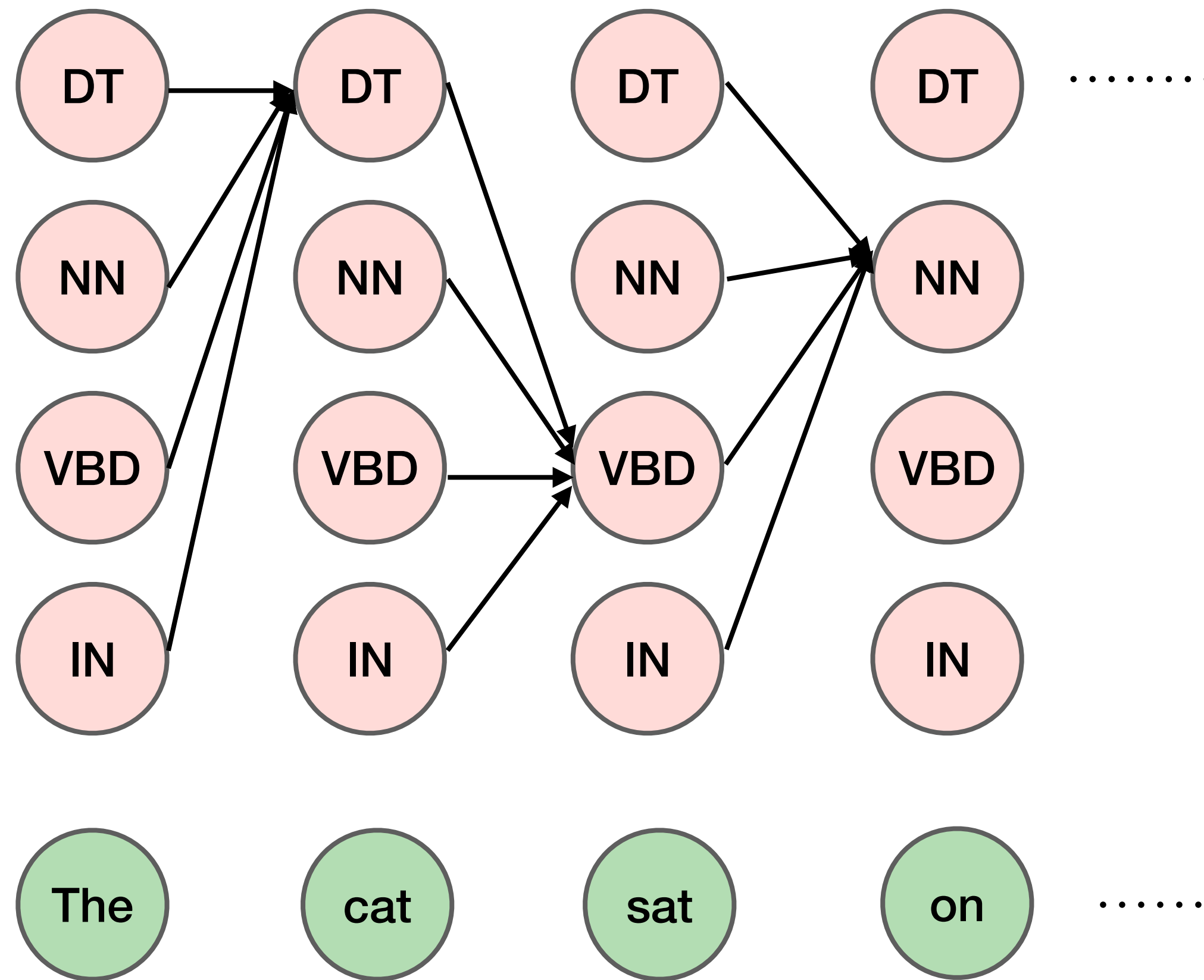


$$M[2, NN] = \max_k \{ M[1, k] P(NN | k) P(\text{cat} | NN) \}$$

$$M[2, NN] = \max_k \{ M[1, k] + \log P(NN | k) + \log P(\text{cat} | NN) \}$$

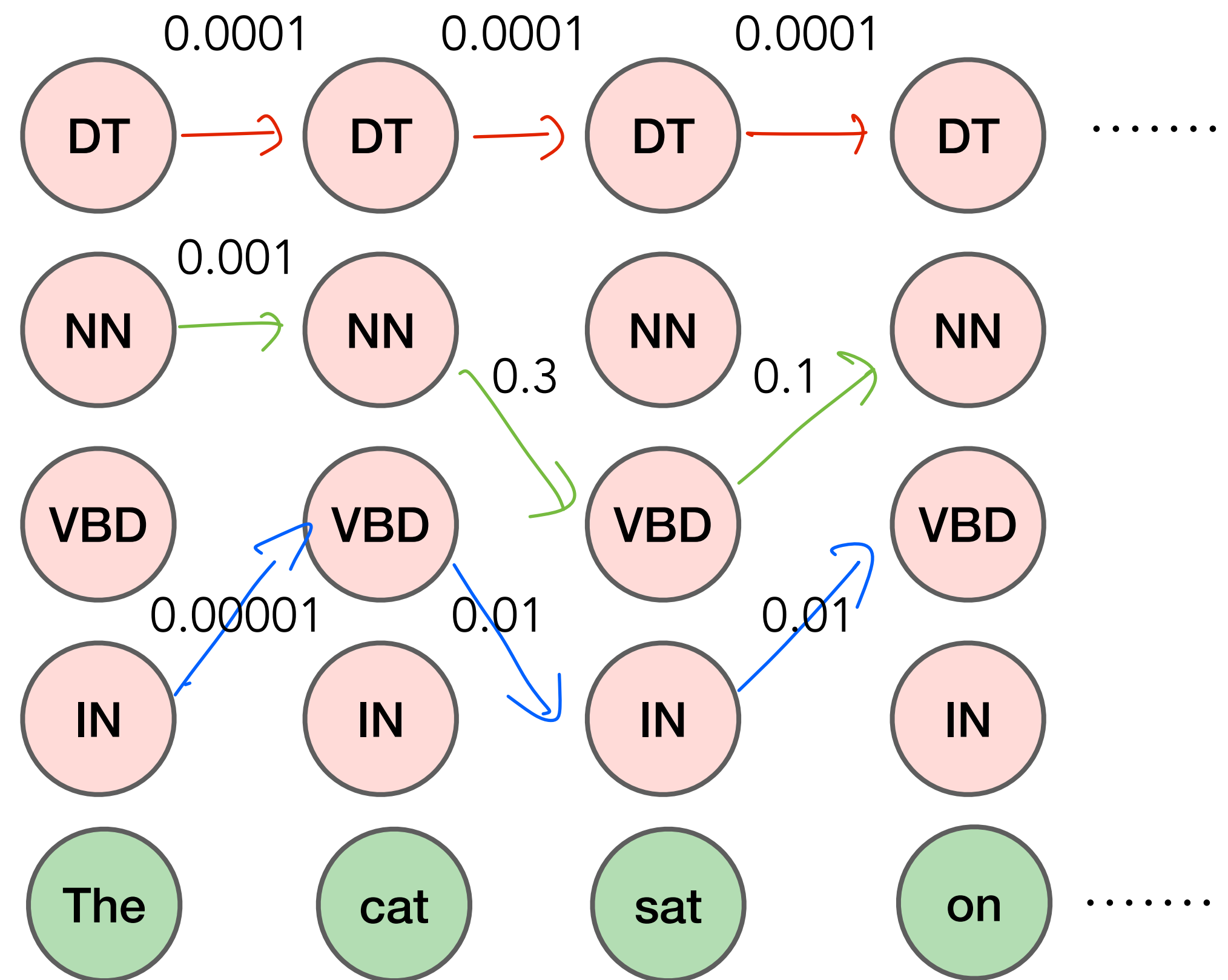
Beam search

If K (number of possible hidden states) is too large, Viterbi is too expensive!



Beam search

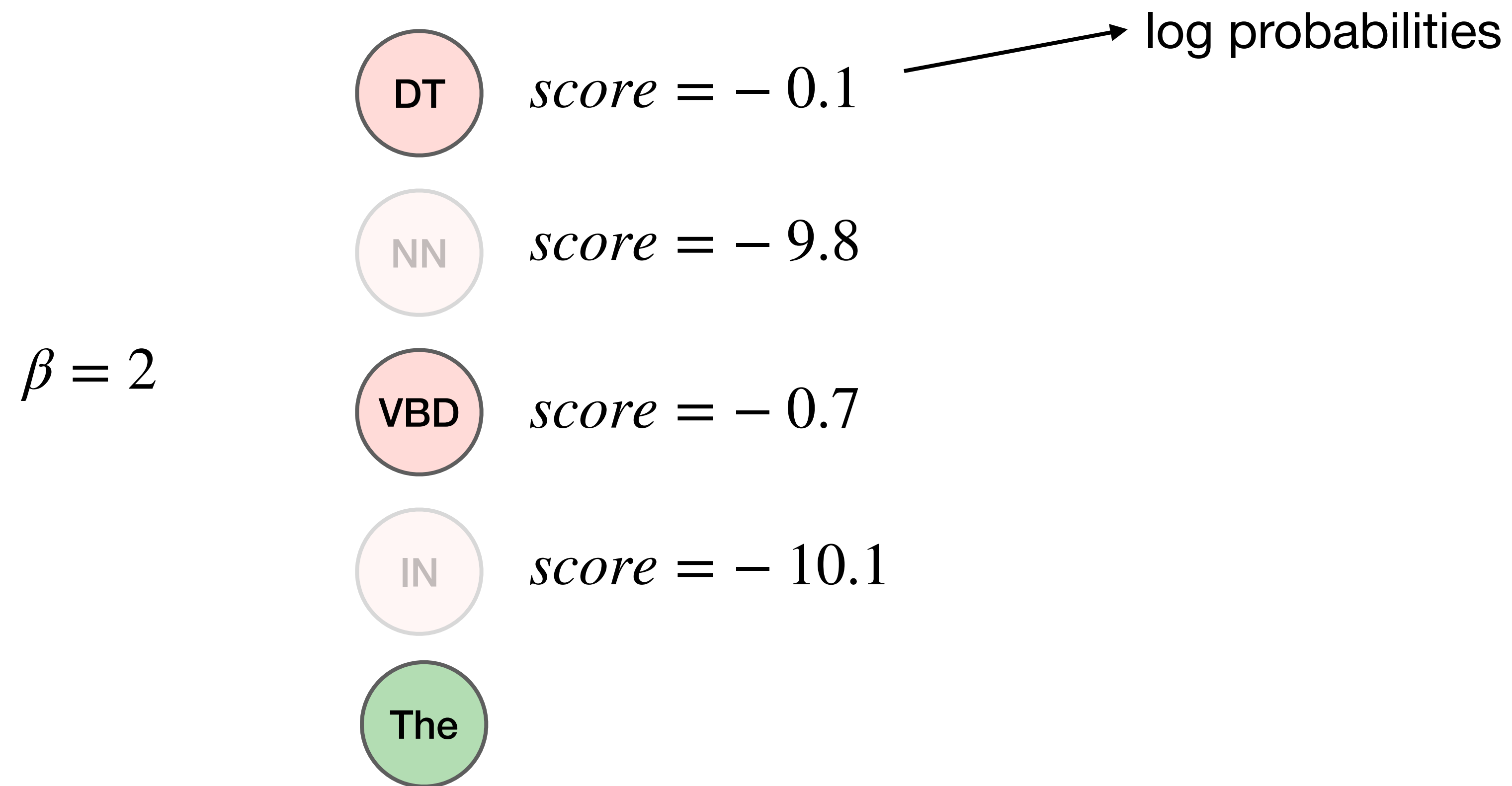
- If K (number of possible hidden states) is too large, Viterbi is too expensive!



Observation: *Many paths have very low likelihood!*

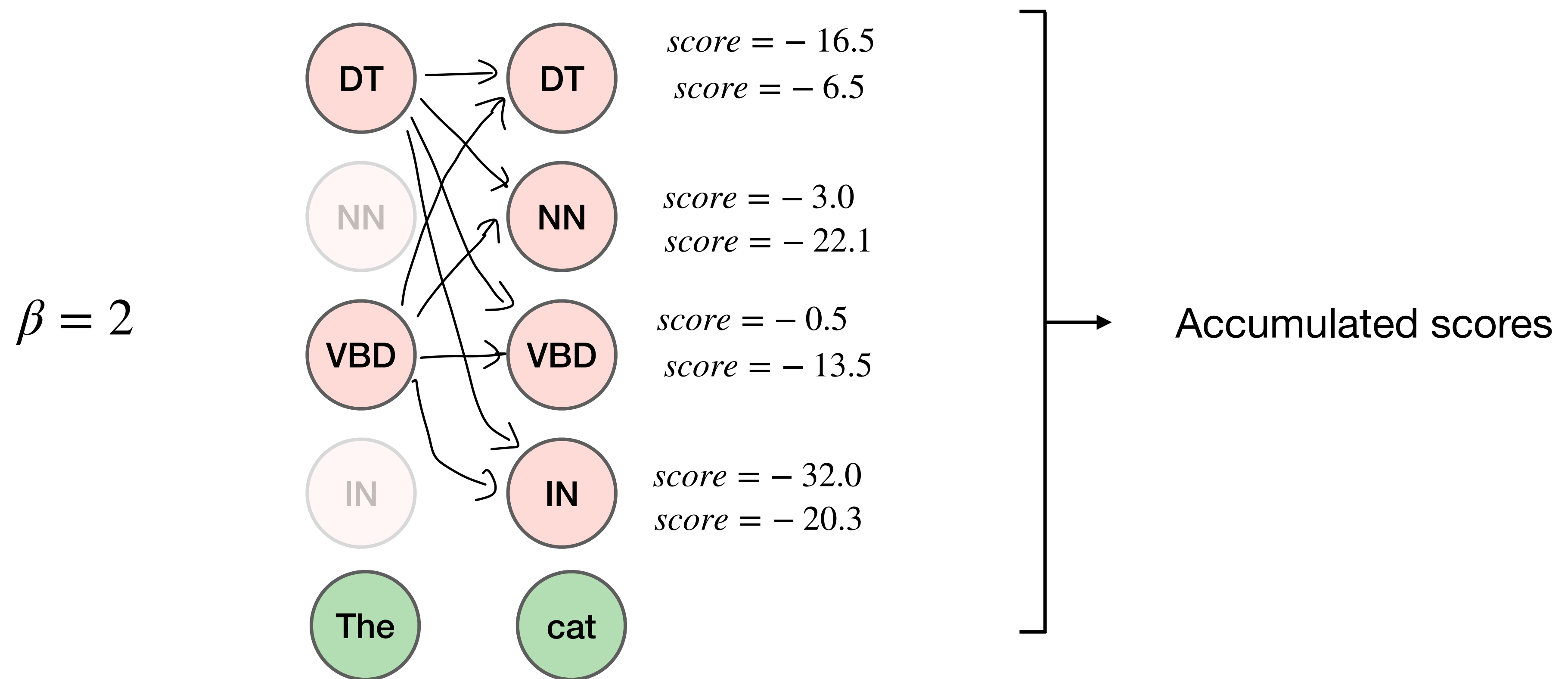
Beam search

- Keep a fixed number of hypotheses at each point
- Beam width, β



Beam search

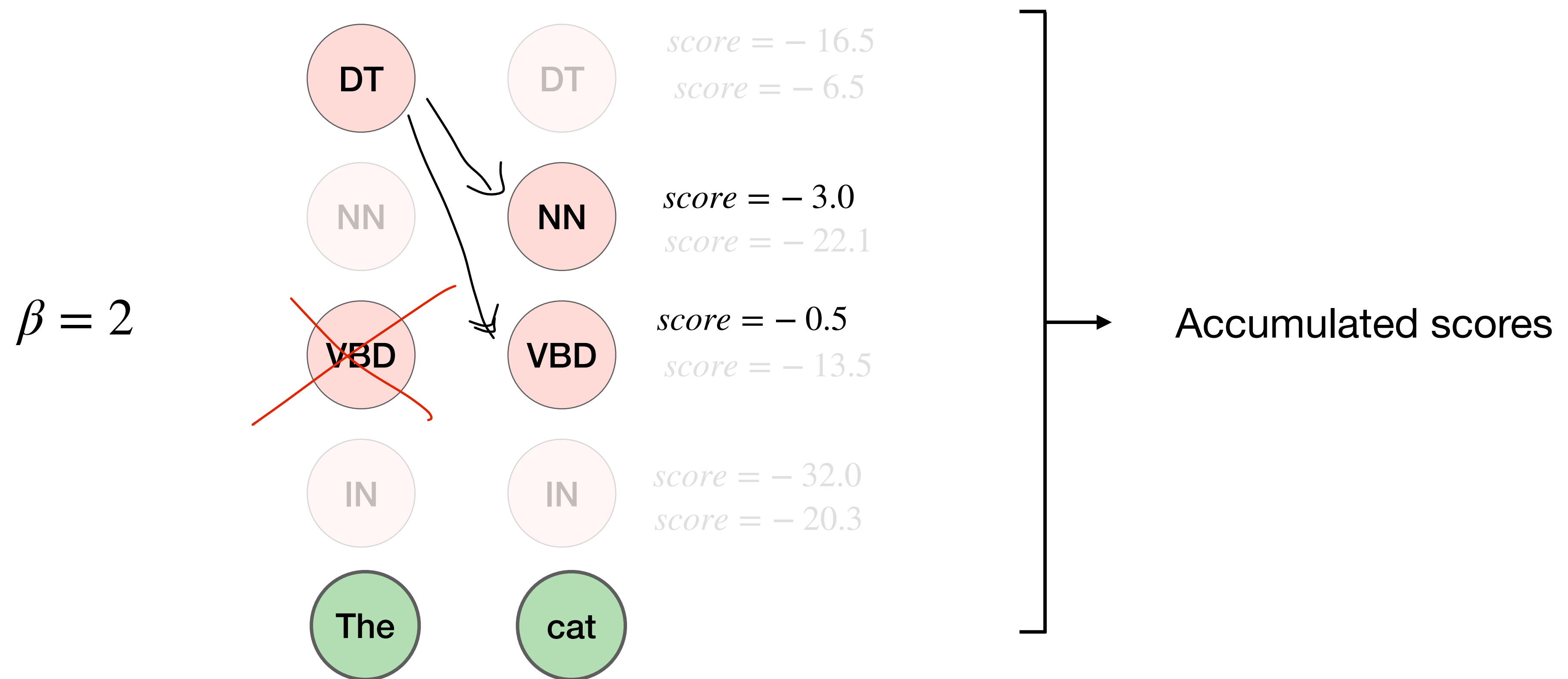
- Keep a fixed number of hypotheses at each point
- Beam width, β



Step 1: Expand all partial sequences in current beam

Beam search

- Keep a fixed number of hypotheses at each point
- Beam width, β

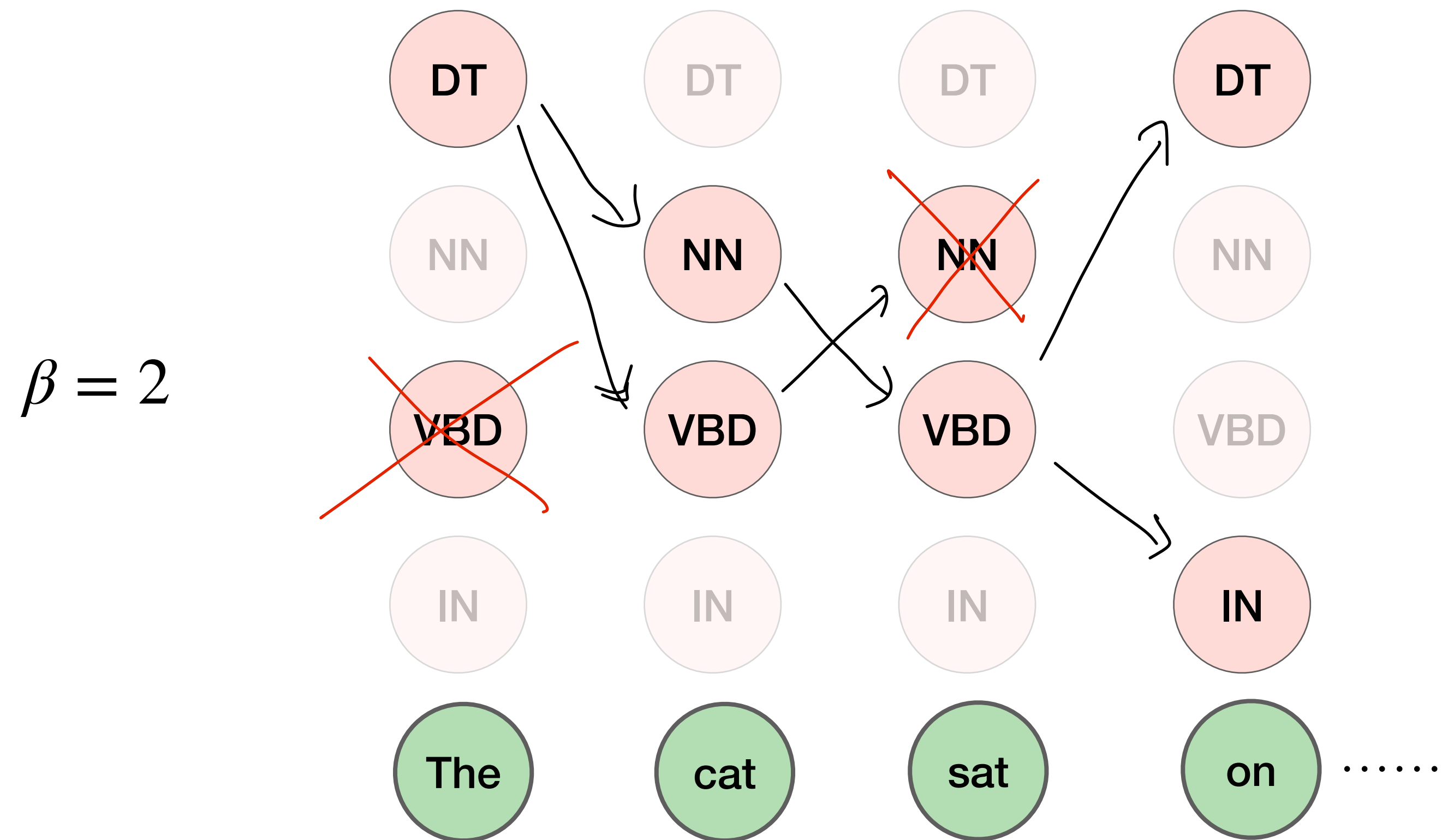


Step 2: Prune set back to top β sequences (sort and select)

... and Repeat!

Beam search

- Keep a fixed number of hypotheses at each point
- Beam width, β



What is the time complexity of this algorithm?

n = number of timesteps
 K = number of states
 β = beam width

A: $O(nK\beta)$

Pick $\max_k M[n, k]$ from within beam and backtrack

Beam Search

- If K (number of states) is too large, Viterbi is too expensive!
- Keep a fixed number of hypotheses at each point
 - Beam width, β
- Trade-off (some) accuracy for computational savings
- **Final remark:** beam search is a common decoding method for any language generation tasks (e.g., n-gram LMs, GPT-3)

Greedy: choose the most likely word!

To predict the next word given a context of two words w_1, w_2 :

$$w_3 = \arg \max_{w \in V} P(w | w_1, w_2)$$