NUS | Computing
National University of Singapore

CS4248: Natural Language Processing

Lecture 7 — Sequences

Announcements

Deadlines: **all** 14 Mar, 23:59

1. Assignment 2
   - Goal: practice manual feature engineering.
   - To be fair, only certain technologies already covered are allowed.
2. Midterm Survey — 1% of your course marks (5 mins of your time)
3. Project: TEAMMATES Intra-Project Formative Feedback
   - (ungraded, but monitored)

4. Correction: Cross Entropy (Slide 34, Lecture 5): -clean version updated.

Recap of Week 06

Term–Context Matrix — Toy Example

...has shown that the movie rating reflects to overall quality...
...the cast of the show turned in a great performance and...
...is to get nlp data for ai algorithms on a large scale...
...only with enough data can ai find reliable patterns to be effective…
…

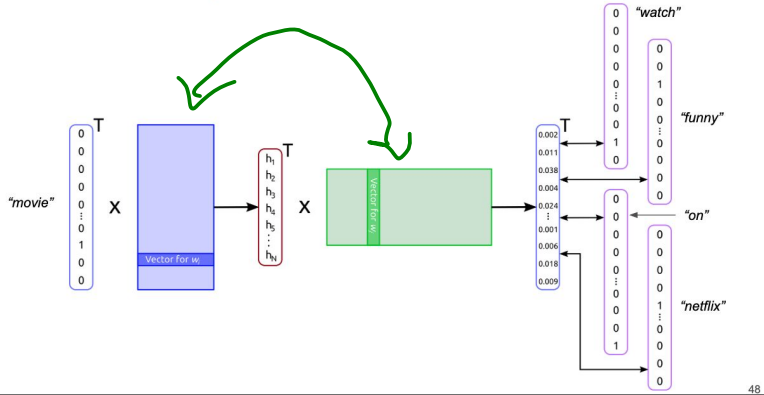
	aardvark	rating	story	data	cast	result	...
movie	0	2	4	0	1	0	
show	0	6	3	0	2	1	
nlp	0	0	1	3	0	4	
ai	0	1	0	5	0	2	

movie ≈ show

nlp ≈ ai

28

Word2Vec — Skip-Gram (window size m=2)

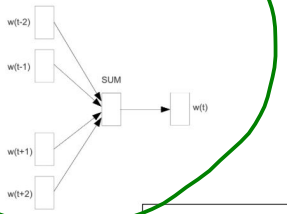


48

Word2Vec: CBOW & Skip-Gram

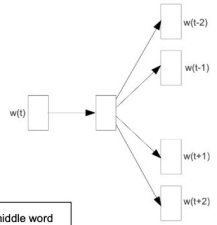
Continuous Bag of Words (CBOW)

Given context ➜ Predict word



Skip-gram

Given word ➜ Predict context



Context = window of words surrounding the middle word

Source: Exploiting Similarities among Languages for Machine Translation

38

Towards Debiasing — Embedding Transformation

Frobenius norm:

$$\|A\|_F^2 = \sqrt{\sum_{i=1}^{} \sum_{j=1} |a_{ij}|^2}$$

- Goal: Transform embeddings to remove gender bias
  - Idea: Find a transformation matrix T the transforms the embedding matrix W
  - Approach: Find T by minimizing

$$\min_T \|(TW)^\top (TW) - (W^\top W)\|_F^2 + \lambda \|(TN)^\top (TB)\|_F^2$$

- $(TW)^\top (TW)$: inner products of embeddings **after** transformation
- $(W^\top W)$: inner products of embeddings **before** transformation
- $(TN)^\top$: transformed gender-neutral word vectors
- (TB): transformed gender subspace

Keep difference (here: Frobenius norm) small — preserve the original embeddings as much as possible!

Minimal if gender subspace removed from vectors of gender-neutral words

Source: Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

88

Pre-Lecture Activity for Next Week

- Assigned Task (due before Mar 8)
  - Post a 1-2 paragraph answer to the following question in the [Pre-Lecture] discussion
  - Watch the panel discussion of the recent **Generative AI, Free Speech, & Public Discourse** https://www.youtube.com/live/BBhewsinQwU?si=ODkIpYjqOCLZh8xD&t=8659

Relate an opinion by one of the panelists to the lecture materials presented today. Why did you pick this opinion to highlight and what is your own opinion on it?

Side notes:

- We will talk about this in the next lecture
- You can just copy-&-paste others' answers or use AI Tools, but please consider your original stance and opinion too.

"It's probably more of a societal problem that needs to be addressed." This was brought up as a concern for the society where as generative AIs become more accessible to public, it becomes easier for people to consume media (whether it is text, image or videos, music even) which are purely tailored to their bias.

During lecture, we have covered NLP Ethics, specifically how natural language is not just a system of words and meanings, but rather interaction of humans and meanings. Because humans have biases, models trained on their interaction will show the same bias. If these are not accounted for properly as we build our models, they will produce biased output and encourage biased human interaction, which could also be fed to future models, and the bias will get amplified over time as a result. People will end up living in their own bubble and will be living a very different world from others.

Now is it really a bad thing? What is wrong with seeing only what you want? That is why the panelist mentioned it is more of a societal problem. It is not socially wrong to maximize your own utility, which is why there is no strong motivation to debias models, worsened by the fact that debiasing is not a trivial task. Your YouTube video recommendation algorithm is a strong example of this. However, we have seen the long term consequences of inbreeding in nature which makes a species vulnerable to diseases and recessive traits. I believe similar consequence will appear in the society in the long run if biasness is not handled correctly, resulting in killing/pruning of brilliant ideas that could have sprouted from stochastic interaction. Fortunately, though debiasing is hard, detecting/identifying bias is relatively easy (e.g. sentiment analysis). This means it is not impossible to come up with a standard measurement for biasness in models that can help regulate generative AIs to slow down division in society.

In the discussion, Kathy McKeown mentioned that AI summarization of articles in a particular domain can be highly accurate, yet this does not necessarily apply to other domains or other type of summarizations (e.g. conversations/dialogue). An AI model is not only prone to making mistakes in its summarization, but may also mischaracterize or misinterpret information in the source material, and often these mistakes are very hard to spot.

In today's age of information overload, especially given the influx of AI-generated content, summarization can be a useful tool to quickly gather sentiments, arguments, opinions, facts and other information on various topics. However, in the process of summarization, the models can inject bias into otherwise bias-free information (e.g. bias in word embeddings as in lecture 6) and potentially present false or inaccurate information. Not only so, summarization may also oversimplify complex or nuanced issues, reducing the quality of the author's arguments or making it one-sided. If a person reading the summarization believes that the inaccurate information in the summary is the true opinion being expressed by the author, it could lead to misunderstandings and even misinformation being spread about the author.

Outline

- **Overview: Sequence Tasks**
- POS Tagging
  - What are Parts of Speech?
  - Why is this task important and challenging?
- Hidden Markov Models (HMM)
  - Basic setup and components
  - Core HMM tasks
    - Model Learning
    - Likelihood computation
    - Viterbi decoding

Motivation

- So far: Bag-of-Words (BoW) models
  - Bag = whole document (e.g., Naive Bayes, Vector Space Model)
  - Bag = context of a word (e.g, PPMI and Word2Vec embeddings)

- Natural language: word order matters! (can vary greatly between languages, though)

Bob kills mosquitoes using the book of Hamlet

vs.

Hamlet kills Bob using the book of mosquitoes

The food tastes good and does not look bad

vs.

The food tastes bad and does not look good

Same words, very different meanings!

Motivation — Example: English

- Fundamental rules word order
  - Subject—Verb—Object (SVO)
  - Adjectives only (immediately) before nouns
  - …and many more

- “Informal” rules — e.g.: order of adjectives
  - Rule: opinion→size→physical quality→shape→age→color→origin→material→type→purpose

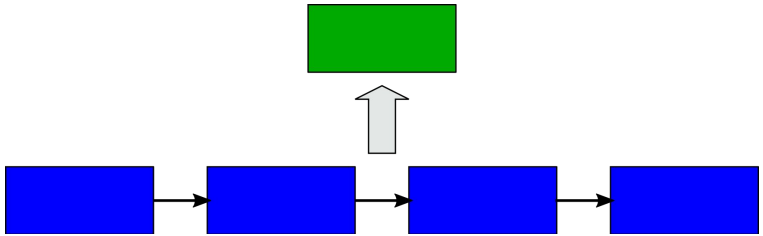
I saw a beautiful, old, blue, German car.

vs.

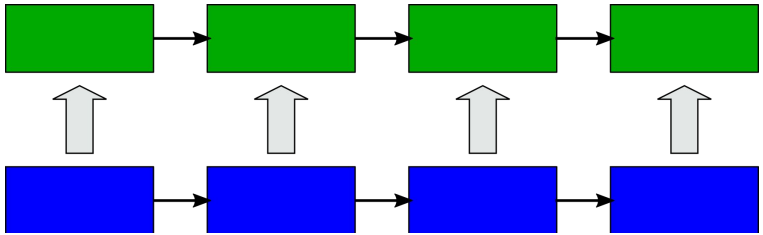
I saw a German, blue, beautiful, old car.

Types of Sequence Tasks

- Sequence classification
  (Many-to-One, N→1)
  - Sentiment analysis
  - Document categorization

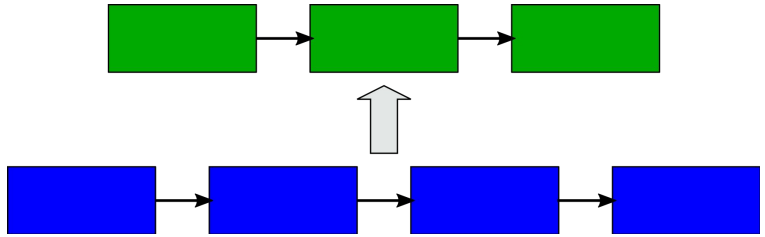


- Sequence labeling/tagging
  (Many-to-Many, N→N)
  - Part-of-Speech Tagging
  - Named Entity Recognition

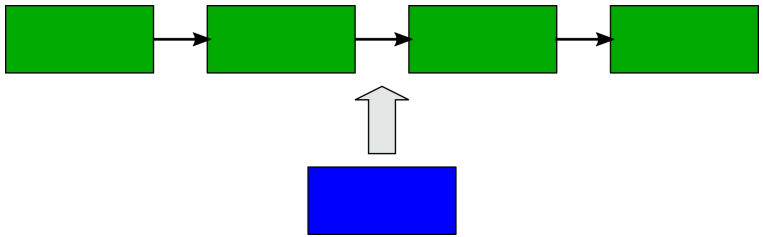


Types of Sequence Tasks

- Sequence translation
  (Many-to-One, N→M)
  - Machine translation
  - Sentence simplification
  - Text summarization



- Sequence generation
  (One-to-Many, 1→N)
  - Image captioning

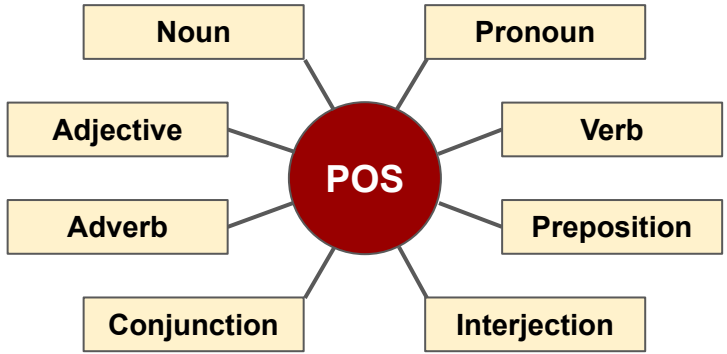


Outline

- Overview: Sequence Tasks
- **POS Tagging**
  - **What are Parts of Speech?**
  - Why is this task important and challenging?
- Hidden Markov Models (HMM)
  - Basic setup and components
  - Core HMM tasks
    - Model Learning
    - Likelihood computation
    - Viterbi decoding

Part-of-Speech Tagging

- Part of Speech (also: word class or syntactic category)
  - Each word belongs one or more of these classes
  - English: 8 main parts of speech / word classes
    (many additional classes and subclasses considered in practice)



- Part-of-Speech (POS) tagging
  - Assign each word in a text a part of speech (duh!)

proper noun singular	verb past tense	adverb	preposition or subordinating conjunction		possessive pronoun	adjective	noun, singular or mass	punctuation
NNP	**VBD**	**RB**	**IN**	**IN**	**PRP$**	**JJ**	**NN**	**.**
Bob	*walked*	*slowly*	*because*	*of*	*his*	*swollen*	*ankle*	*.*

Penn Treebank Tagset

Base set: 36 POS tags

CC	Coordinating conjunction	*and, or*
CD	Cardinal number	*1, 2, 3, one, two, three*
DT	Determiner	*the, a, an, any, some*
EX	Existential there	
FW	Foreign word	
IN	Preposition / subord. conjunction	*in, into, whether, if*
JJ	Adjective	*cleaner, nice*
JJR	Adjective (comparative)	*cleaner, nicer*
JJS	Adjective (superlative)	*cleanes, nicest*
LS	List item marker	
MD	Modal	*can, could, may*
NN	Noun (singular or mass)	*machine, computer, air*
NNS	Noun (plural)	*machines, computers*
NNP	Proper noun (singular)	*Clementi Mall*
NNPS	Proper noun (plural)	*Americas*
PDT	Predeterminer	*all, both, half*
POS	Possessive ending	*'s*
PRP	Personal pronoun	*him. himself, we*

PP$	Possessive pronoun	*her, our, ours*
RB	Adverb	*quickly, swiftly*
RBR	Adverb (comparative)	*further, greater, more*
RBS	Adverb (superlative)	*furthest, greatest, most*
RP	Particle	*across, up*
SYM	Symbol	*=, +, &*
TO	to	*to*
UH	Interjection	*shucks, heck, oops*
VB	Verb (base form)	*be, assign, run*
VBD	Verb (past tense)	*was, assigned, ran*
VBG	Verb (gerund / present participle)	*being, assigning*
VBN	Verb (past participle)	*been, assigned*
VBP	Verb (non-3rd pers. sing. present)	*am, are*
VBZ	Verb (3rd pers. sing. present)	*is*
WDT	wh-determiner	*that, which, what*
WP	wh-pronoun	*that, which, whom*
WP$	Possessive wh-pronoun	*whose*
WRB	wh-adverb	*how, however, why*

Penn Treebank Tagset

Extended set: 12 tags for punctuations and special symbols

#	Pound sign	#
$	Dollar sign	$
.	Sentence-final punctuation	. ? !
:	Sentence-middle punctuation	: ; … - —
,	Comma	,
(	Left bracket character	([{ <
)	Right bracket character	)] } >
"	Straight double quote	
‘	Left open single quote	
“	Left open double quote	
’	Right close single quote	
”	Right close double quote	

Part of Speech — Two Broad Categories

- **Closed class** words
  - Small, fixed membership — reasonably easy to enumerate
  - Generally, short function words that “structure” sentences
  - Examples: prepositions, pronouns, participles, determiners, conjunctions, etc.

- **Open class** words
  - Impossible to completely enumerate
  - New words continuously being invented, borrowed, etc.
  - For most languages: nouns, verbs, adjectives, adverbs

Outline

- Overview: Sequence Tasks
- **POS Tagging**
  - What are Parts of Speech?
  - **Why is this task important and challenging?**
- Hidden Markov Models (HMM)
  - Basic setup and components
  - Core HMM tasks
    - Model Learning
    - Likelihood computation
    - Viterbi decoding

POS Tagging — Why is it Important?

- Very useful or crucial for many NLP downstream tasks
  - Named Entity recognition (typically comprised of nouns and proper nouns)
  - Information extractions (e.g., verbs indicate relations between entities)
  - Parsing (information of word classes useful before creating parse trees)
  - Speech synthesis/recognition (e.g., noun “DIScount” vs. verb “disCOUNT”)
  - Authorship Attribution (e.g., relative frequencies of nouns, verbs, adjectives, etc.)
  - Machine Translation (e.g., reordering of adjectives and nouns)

➜ POS tagging: important low-level NLP task

Quick Quiz

Which POS have the highlighted words in the following headlines?

*"Teacher **Strikes** Idle Kids"*

*"Hershey **Bars** Protest"*

A verb / noun

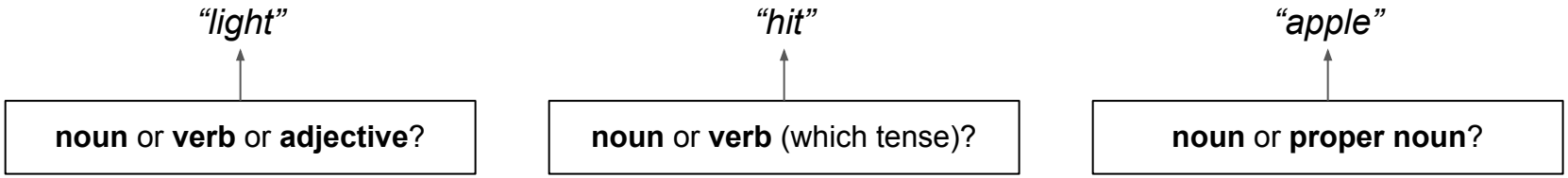
B noun / noun

C verb / verb

D noun / verb

POS Tagging — Why is it Difficult? And How Difficult?

- Our common problem: **Ambiguity**
  - Many common words have multiple meanings ➜ multiple POS



  - Often ambiguous, even with additional context

    (even humans often don't agree on the correct labeling!)

POS Tagging — Why is it Difficult? And How Difficult?

- POS tagging in English
  - ~85% of word types are unambiguous (e.g., “quickly” is always an adverb, “Alice” is always a proper noun)
  - ~15% of word types are ambiguous — but those are quite common!

→ 55-65% of word tokens are ambiguous
  - Ambiguous = 2 or more possible POS tags
  - Results depend on text corpus

Quick Quiz

Which word can be assigned the **most** POS tags?

A *light*

B *up*

C ~~*to*~~

D *bank*

🏃‍♀️🏃🏃‍♂️ Flying Planes Can Be Dangerous

- Activity: Find or construct one or more examples of ambiguous newspaper headlines 😅😂🤣 and explain the ambiguity
  (Bonus Points if its due to POS ambiguity)

- Do keep your headlines non-offensive

- Post your answer to Canvas > Discussions > [In-Lecture Interaction] L1
  (Help like other classmate's responses too! 👍)

POS Tagging — Baseline Algorithm

- Most straightforward approach
  - Label each word with its most frequent POS tag
  - Label unknown words as nouns (most common open world class)

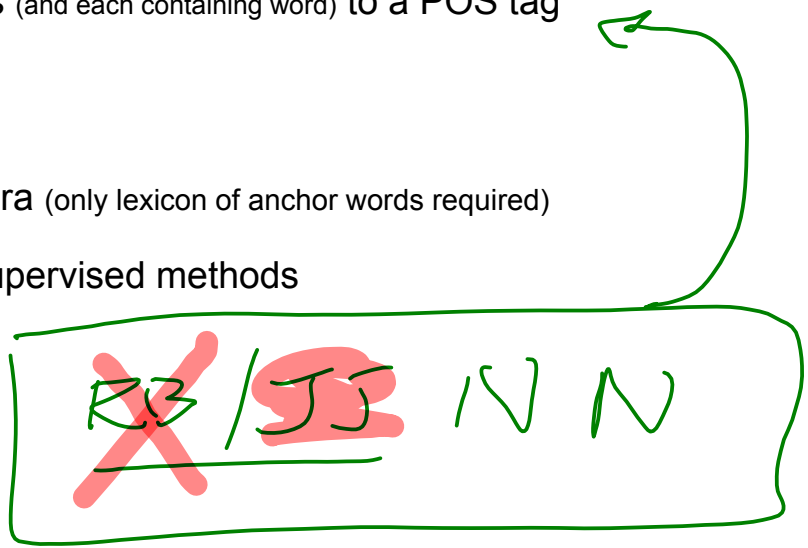
➜ Result: ~92% accuracy (vs. ~97-98% accuracy for SOTA methods)
  - Doesn't sound so bad, right?
  - 2 main problems:

    (1) **Imbalanced errors**
      - High accuracy due to common/frequent unambiguous words (e.g., "the", "a/an", "and", "or")
      - Many of these words also often not that interesting for downstream NLP tasks

    (2) **Downstream error propagation**
      - POS tagging as low-level NLP task ➜ errors quickly propagate up

POS Tagging — Unsupervised Algorithms

- Basic intuition
  - Utilize words with unambiguous POS tags ➜ **anchor words**
  - Observe patterns to group words into clusters of the same word class
  - Use anchor words to assign clusters (and each containing word) to a POS tag
- Practical considerations
  - No need for hand-labeled text corpora (only lexicon of anchor words required)
  - Poorer performance compared to supervised methods



POS Tagging — Supervised Methods

- Require hand-labeled text corpus
  - Used as input training data for supervised models
  - Challenging for low-resource languages
    (i.e., languages lacking in large, annotated datasets)

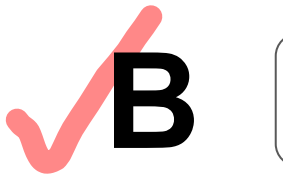
- Popular models (all yielding quite similar results)
  - Hidden Markov Models (HMM)
  - Conditional Random Fields (CRF)
  - Neural sequence models (RNNs, Transformers)
  - Large language models (e.g., BERT)

- Accuracies have reached the human ceiling (i.e., POS taggers as good as human annotators)
- POS tagging considered a solved task for high-resource languages (e.g. English)
- Limitations: low-resource languages and special application domains

Quick Quiz

Which word is **not** an **anchor word**?

(i.e., which word has only 1 POS?)



A *sun*

B *venus*

C *earth*

D *mars*

An Unusual Way of Speaking Yoda Has

In the *Star Wars* series, the character Yoda speaks in an O–S–V convention.

This is a by far the rarest natural language word ordering, limited to a few Amerindian languages indigenous to the Amazon basin.

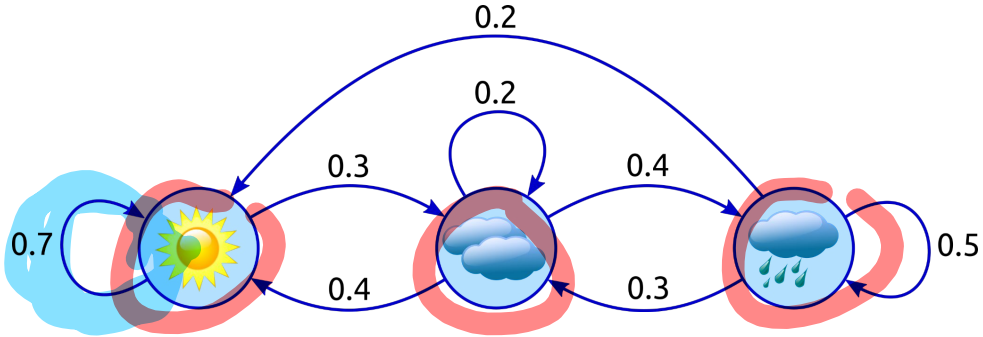
Image Credits: Riku Lu @ Unsplash

Outline

- Overview: Sequence Tasks
- POS Tagging
  - What are Parts of Speech?
  - Why is this task important and challenging?
- **Hidden Markov Models (HMM)**
  - **Basic setup and components**
  - Core HMM tasks
    - Model Learning
    - Likelihood computation
    - Viterbi decoding

Markov Chains

- Markov Chain
  - Models transitions between a set of states using transition probabilities (captured by a transition matrix A)
  - Transition only depends on current state (Markov assumption)
  - Sequence: series of transitions



- Example: "Daily Weather"
  - 3 states: *sunny*, *cloudy*, *rainy*

$$A = \begin{bmatrix} 0.7 & 0.3 & 0.0 \\ 0.4 & 0.2 & 0.4 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

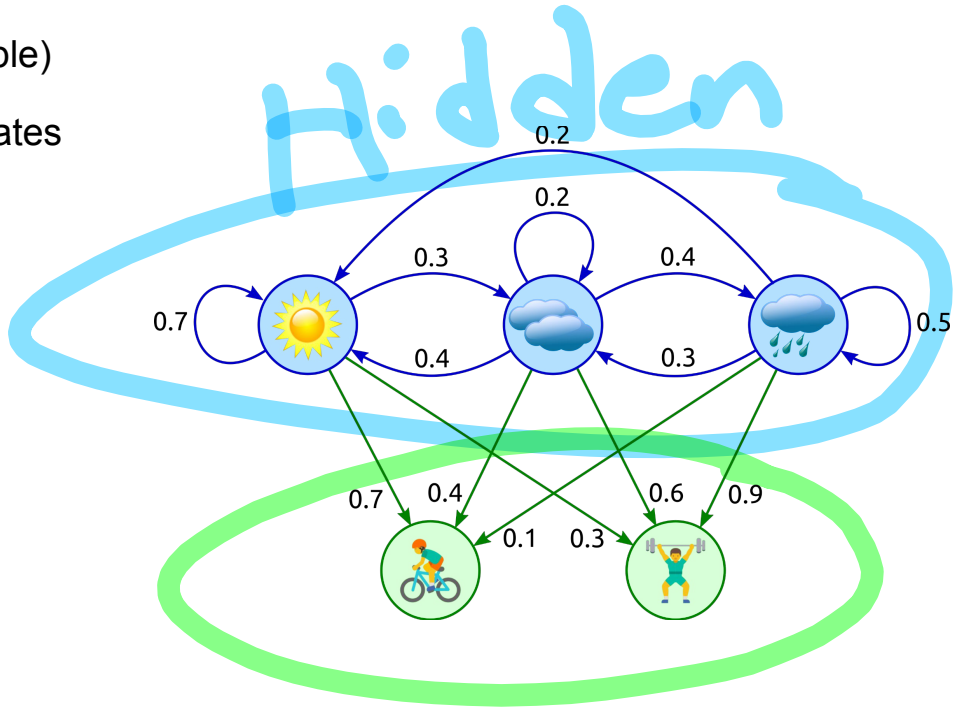
Example question: *"What is the probability of getting a 5 sunny days in a row?"*

Markov Chain → Hidden Markov Models

- Hidden Markov Models (HMM)
  - States are hidden (i.e., not directly observable)
  - Observable variables that depend on the states

- Example: Exercise Routine
  - 3 hidden(!) states: *sunny*, *cloudy*, rainy
  - 2 observed activities: *biking*, *lifting*
    (with the activity depending on the weather)

Example question: *"Given that Chris spent the first 3 days lifting and then 3 days biking, what was the most likely weather over the last 6 days?"*



HMM — Components

1. Finite Set of **states** $S = \{s_1, s_2, ..., s_N\}$

2. Sequence of states
$Q = q_1, q_2, q_3, \ldots, q_T$, with $q_t \in S$

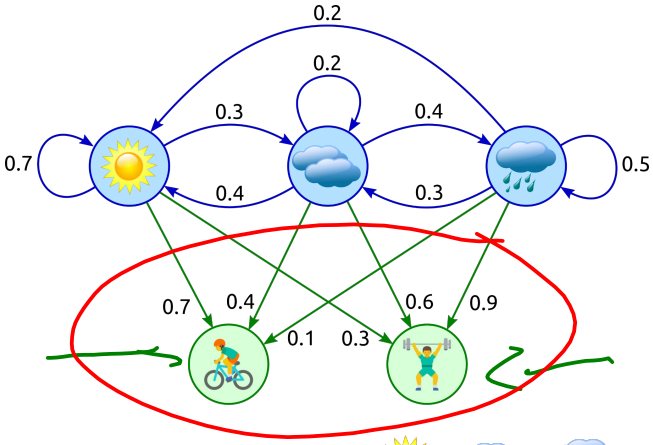
3. Finite set of **symbols** $V = \{v_1, v_2, ..., v_M\}$

3. Sequence of observations
$O = o_1, o_2, o_3, \ldots, o_T$, with $o_t \in V$

4. **Transition probability matrix** A
$A = \{a_{ij}\}, \quad a_{ij} = P(q_{t+1} = s_j | \ q_t = s_i)$

5. **Observation / emission probability matrix** B
$B = \{b_i(o_k)\}, \quad b_i(o_k) = P(o_t = v_k | \ q_t = s_i)$

6. **Initial state distribution** π
$\pi = \{\pi_i\}, \quad \pi_i = P(q_1 \mid s_i)$



$$A = \{a_{ij}\} = \begin{bmatrix} 0.7 & 0.3 & 0.0 \\ 0.4 & 0.2 & 0.4 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

$$B = \{b_i(o_k)\} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \\ 0.1 & 0.9 \end{bmatrix}$$

HMM — Probabilities (annotated)

i is a starting state
j is a destination state

Transition probability matrix A

$$A = \{a_{ij}\}, \quad a_{ij} = P(q_{t+1} = s_j | \ q_t = s_i)$$

probability of moving from state i to state j

Probability of transitioning from state s_i to s_j at any time t

$$\sum_{j}^{N} a_{ij} = 1 \quad \forall i$$

Observation / emission probability matrix B

$$B = \{b_i(o_k)\}, \quad b_i(o_k) = P(o_t = v_k | \ q_t = s_i)$$

probability of observation v_k given i'm in state s_i

Probability of state s_i generating output v_k at any time t

$$\sum_{k}^{M} b_i(o_k) = 1, \quad \forall k$$

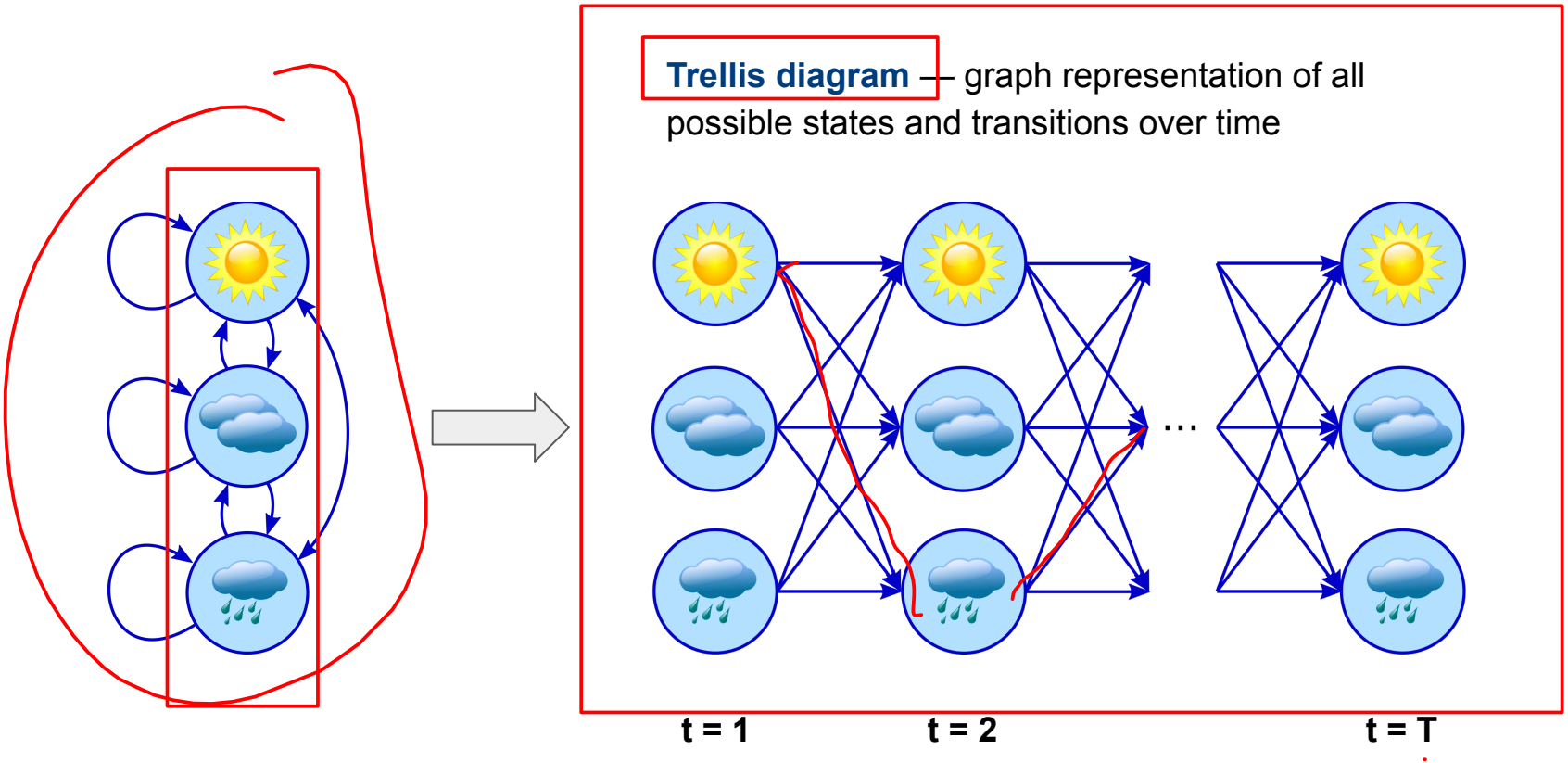
Initial state distribution π

$$\pi = \{\pi_i\}, \quad \pi_i = P(q_1 = s_i)$$

Probability of sequence starting in state s_i

$$\sum_{i}^{N} \pi_i = 1$$

HMM — Unrolled Representation



Quick Quiz

Is the Hidden Markov Model a **generative** model or a **discriminative** model?

A Generative

B Discriminative

C Both

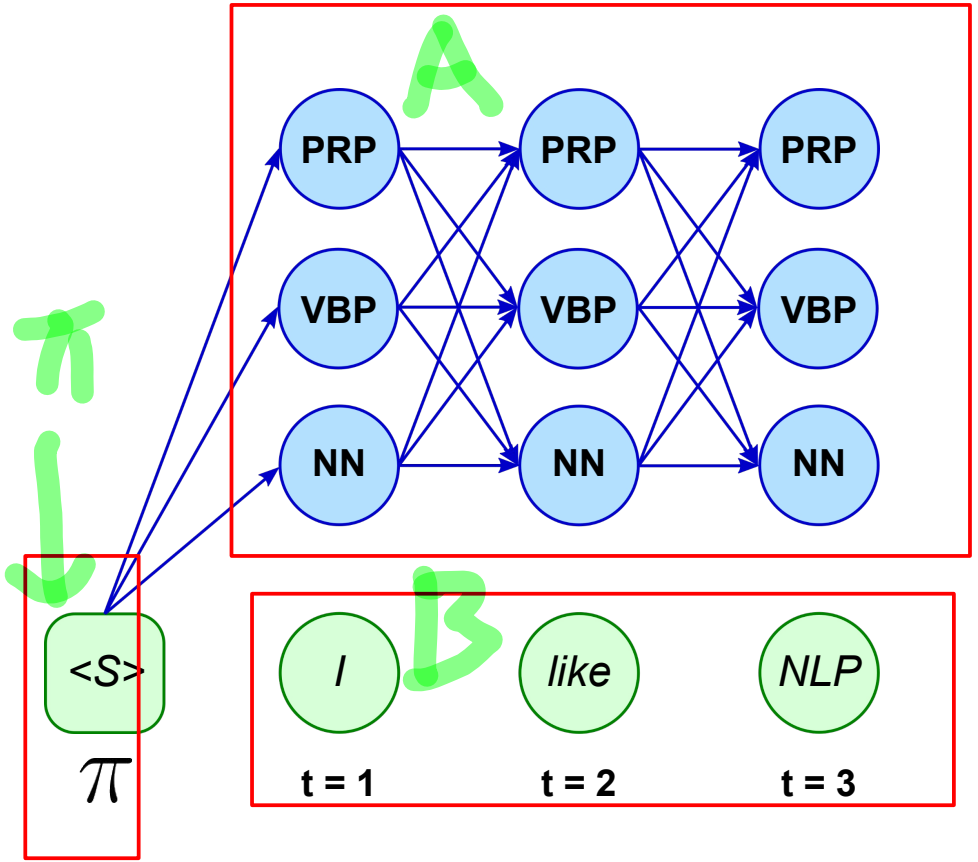
D Neither

Outline

- Overview: Sequence Tasks
- POS Tagging
  - What are Parts of Speech?
  - Why is this task important and challenging?
- **Hidden Markov Models (HMM)**
  - Basic setup and components
  - **Core HMM tasks**
    - ➢ Model Learning
    - ➢ Likelihood computation
    - ➢ Viterbi decoding

POS Tagging with HMMs

- Basic setup
  - Hidden states ➜ POS tags
  - Observations ➜ words
- Example
  - 3 states: {PRP, VBP, NN}



HMM — Core Tasks

(1) Model Learning

Given corresponding state and observation sequences Q and O

➜ Learn all model parameters, i.e., probabilities A, B and π

Training using an annotated dataset

(2) Likelihood

Given an HMM $\theta = (A, B, \pi)$

+ an state sequence Q

+ an observation sequence O

➜ Calculate the probability $P(Q, O|\theta)$

Given 2 POS tag sequences for a sentence, compare which is more likely

(3) Decoding

Given an HMM $\theta = (A, B, \pi)$ + an observation sequence O

➜ Find the most likely sequence of states

Given a sentence, find the most likely POS tags

Outline

- Overview: Sequence Tasks
- POS Tagging
  - What are Parts of Speech?
  - Why is this task important and challenging?
- **Hidden Markov Models (HMM)**
  - Basic setup and components
  - **Core HMM tasks**
    - ➢ **Model Learning**
    - ➢ Likelihood computation
    - ➢ Viterbi decoding

HMM — Model Learning

Quick Quiz: Spot a familiar issue? How can we address it?

- Calculating probabilities using Maximum Likelihood Estimates

$$\pi_i = P(q_1 = s_i) = \frac{Count(\langle S \rangle s_i)}{Count(\langle S \rangle)}$$

sentences starting with state s_i

sentences

$$a_{ij} = P(q_{t+1} = s_j | \, q_t = s_i) = \frac{Count(s_i s_j)}{Count(s_i)}$$

occurrences of state s_i followed by state s_j

occurrences of state s_i

$$b_i(o_k) = P(o_t = v_k | \, q_t = s_i) = \frac{Count(v_k, s_i)}{Count(s_i)}$$

occurrences of observation v_k in state s_i

occurrences of state s_i

HMM — Model Learning — Side Note

- POS tagging using HMM
  - Full supervised task ➜ corpus of words labeled with all the correct POS tags
  - Fully “visible” Markov Model
    (we have the state and observation sequences)

“Direct” parameter learning using MLE
(we just need simple counts)

- Often in other applications
  - State sequences Q are not known
  - Impossible to compute simple counts

Outline

- Overview: Sequence Tasks
- POS Tagging
  - What are Parts of Speech?
  - Why is this task important and challenging?
- **Hidden Markov Models (HMM)**
  - Basic setup and components
  - **Core HMM tasks**
    - ~~Model Learning~~
    - **Likelihood computation**
    - Viterbi decoding

Likelihood

- Given: HMM $\theta = (A, B, \pi)$ and

  $O = o_1, o_2, o_3, \ldots, o_T$

  $Q = q_1, q_2, q_3, \ldots, q_T$

- Calculate joint probability $P(O, Q|\theta)$

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

with $P(q_1|q_0) = P(q_1)$

emission probabilities

transition probabilities

initial state probability

Likelihood — Example

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

- Given: sentence *O*, POS tags *Q*

$O = I, like, NLP$

$Q = PRP, VBN, NN$

$P(\text{“}I, like, NLP\text{”}|PRP-VBN-NN) = ?$

?? , ?? , ??

$$P(\text{“}I, like, NLP\text{”}|PRP - VBN - NN) = P(I|PRP) \cdot P(PRP|\langle S \rangle) \cdot \quad \text{i=1}$$

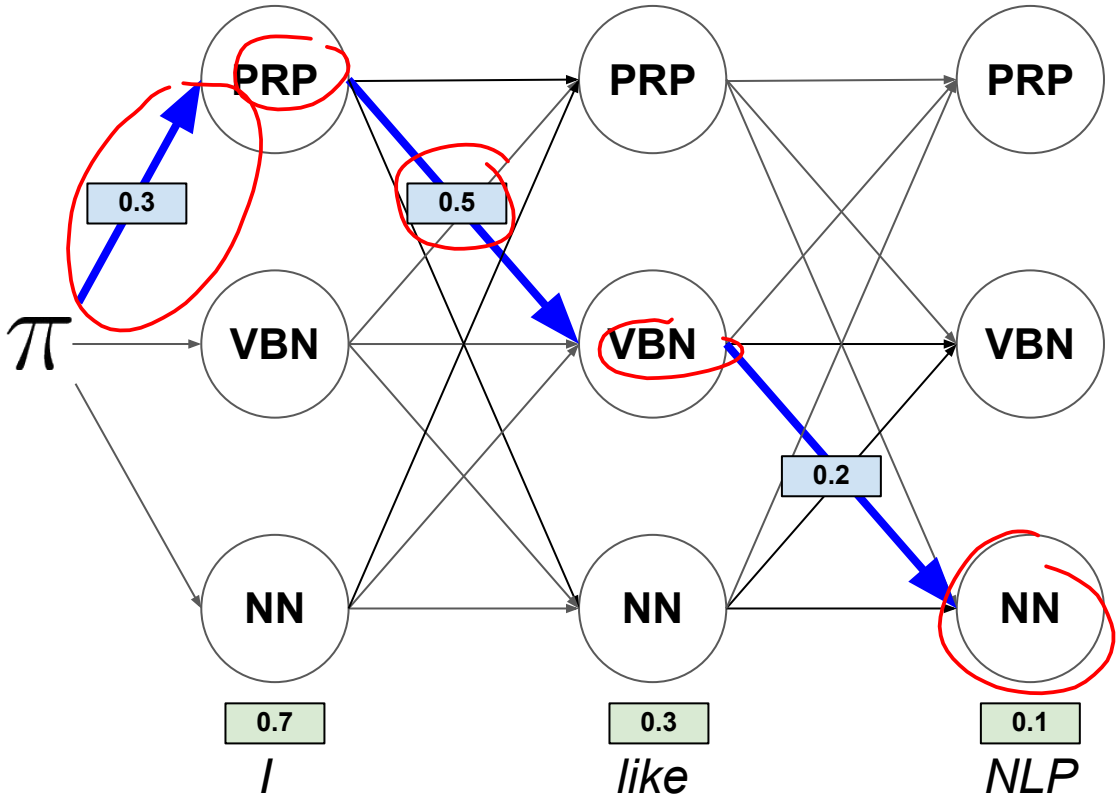
$$P(like|VBN) \cdot P(VBN|PRP) \cdot \quad \text{i=2}$$

$$P(NLP|NN) \cdot P(NN|VBN) \quad \text{i=3}$$

All values can be directly taken from A, B, and π

Likelihood — Example

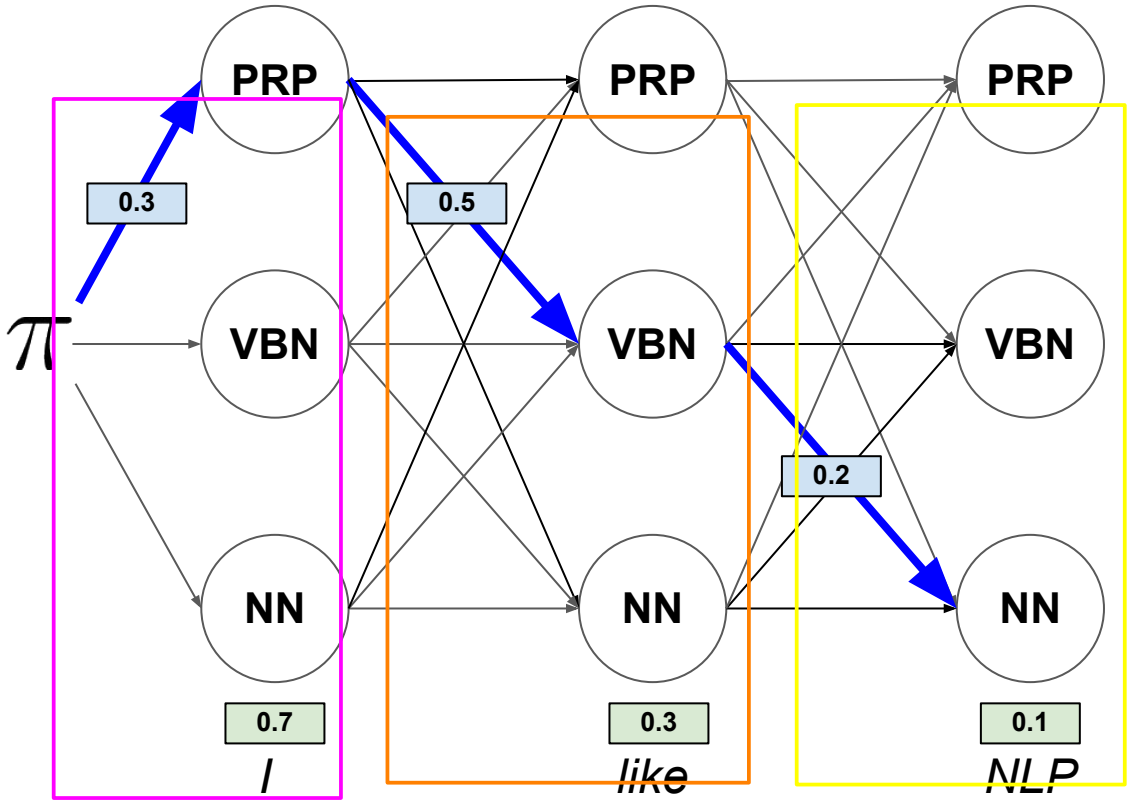
Visualization using a Trellis diagram ➜ just follow the path



$$P(\text{“}I, like, NLP\text{”}|PRP - VBN - NN) =$$
$$P(I|PRP) \cdot P(PRP|\langle S \rangle) \cdot$$
$$P(like|VBN) \cdot P(VBN|PRP) \cdot$$
$$P(NLP|NN) \cdot P(NN|VBN)$$

Likelihood — Example

Visualization using a Trellis diagram ➜ just follow the path

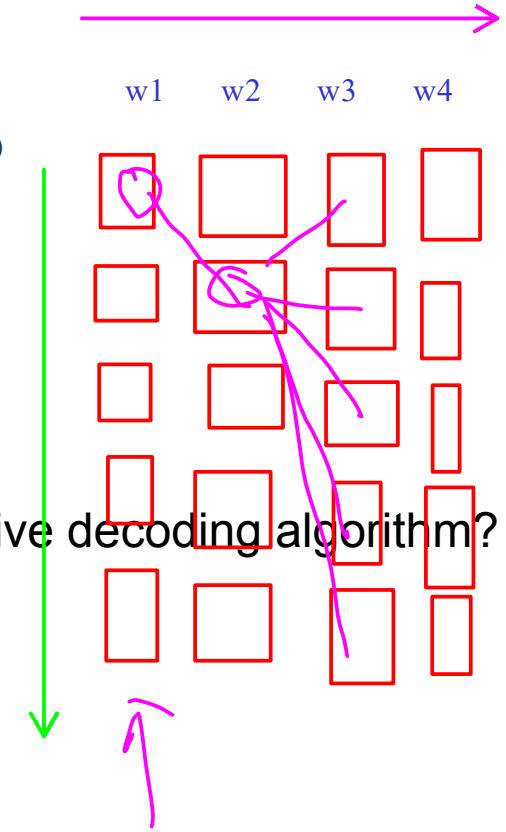


$$P(\text{“}I, like, NLP\text{”}|PRP-VBN-NN) = P(I|PRP)\cdot P(PRP|\langle S\rangle)\cdot P(like|VBN)\cdot P(VBN|PRP)\cdot P(NLP|NN)\cdot P(NN|VBN)$$

$$P(\text{“}I, like, NLP\text{”}|PRP-VBN-NN) = 0.7\cdot 0.3\cdot 0.3\cdot 0.5\cdot 0.1\cdot 0.2 = 0.00063$$

How Complex Could It Be?

- Naive algorithm for decoding (for a given observation sequence O)
  - Enumerate all possible state sequences Q
  - Compute all joint probabilities $P(O, Q)$
  - Return state sequence Q with highest joint probability



What is the **runtime complexity** of this naive decoding algorithm?

3^3 S^N

Outline

- Overview: Sequence Tasks
- POS Tagging
  - What are Parts of Speech?
  - Why is this task important and challenging?
- **Hidden Markov Models (HMM)**
  - Basic setup and components
  - **Core HMM tasks**
    - Model Learning
    - Likelihood computation
    - **Viterbi decoding**

Decoding

- Decoding task
  - Given an HMM $\theta = (A, B, \pi)$ + an observation sequence O
  - Find the most likely sequence of states Q

$$Q = \underset{q_1 \ldots q_t}{\operatorname{argmax}} \prod_{i=1}^{T} P(o_i|q_i) \cdot P(q_i|q_{i-1}) \qquad \text{with } P(q_1|q_0) = P(q_1)$$

emission probabilities

transition probabilities

initial state probability

➜ **Dynamic Programming** to avoid checking all possible state sequences

Viterbi Algorithm — Toy Example

- Oversimplified setup
  - 3 POS tags: **DT** (determiner), **NN** (noun), **VB** (verb)
  - Let's assume the following HMM

Note: The rows in B do not up to 1 since B does not capture all words, only those we needs.

$$\pi = \begin{bmatrix} \textbf{DT} & \textbf{NN} & \textbf{VB} \\ 0.8 & 0.2 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} & \textbf{DT} & \textbf{NN} & \textbf{VB} \\ \textbf{DT} & 0 & 0.8 & 0.2 \\ \textbf{NN} & 0 & 0.5 & 0.5 \\ \textbf{VB} & 0.5 & 0.5 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} & \textit{the} & \textit{fans} & \textit{love} & \textit{show} \\ \textbf{DT} & 0.2 & 0 & 0 & 0 \\ \textbf{NN} & 0.0 & 0.05 & 0.3 & 0.1 \\ \textbf{VB} & 0 & 0.25 & 0.15 & 0.3 \end{bmatrix}$$

- Task: Find the most likely sequence of state (i.e., POS tags) for:

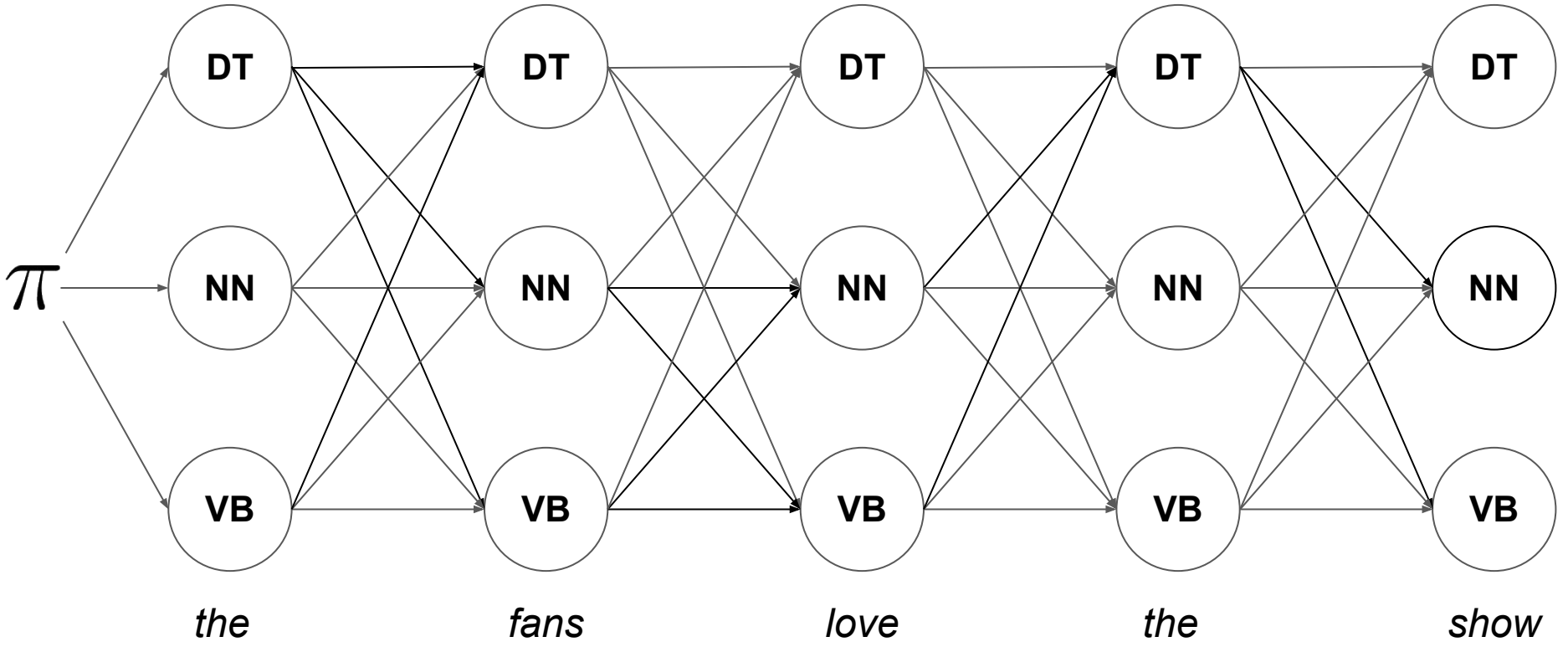
"the fans love the show"

Example

$$\pi = \begin{matrix}\text{DT} & \text{NN} & \text{VB}\end{matrix}\begin{bmatrix}0.8 & 0.2 & 0\end{bmatrix}$$

$$A = \begin{matrix}\text{DT}\\ \text{NN}\\ \text{VB}\end{matrix}\begin{matrix}\overset{\text{DT}}{}\\ \\ \end{matrix}\begin{matrix}\overset{\text{NN}}{}\\ \\ \end{matrix}\begin{matrix}\overset{\text{VB}}{}\\ \\ \end{matrix}\begin{bmatrix}0 & 0.8 & 0.2\\ 0 & 0.5 & 0.5\\ 0.5 & 0.5 & 0\end{bmatrix}$$

$$B = \begin{matrix}\text{DT}\\ \text{NN}\\ \text{VB}\end{matrix}\begin{matrix}\overset{\textit{the}}{}\\ \\ \end{matrix}\begin{matrix}\overset{\textit{fans}}{}\\ \\ \end{matrix}\begin{matrix}\overset{\textit{love}}{}\\ \\ \end{matrix}\begin{matrix}\overset{\textit{show}}{}\\ \\ \end{matrix}\begin{bmatrix}0.2 & 0 & 0 & 0\\ 0.0 & 0.05 & 0.3 & 0.1\\ 0 & 0.25 & 0.15 & 0.3\end{bmatrix}$$

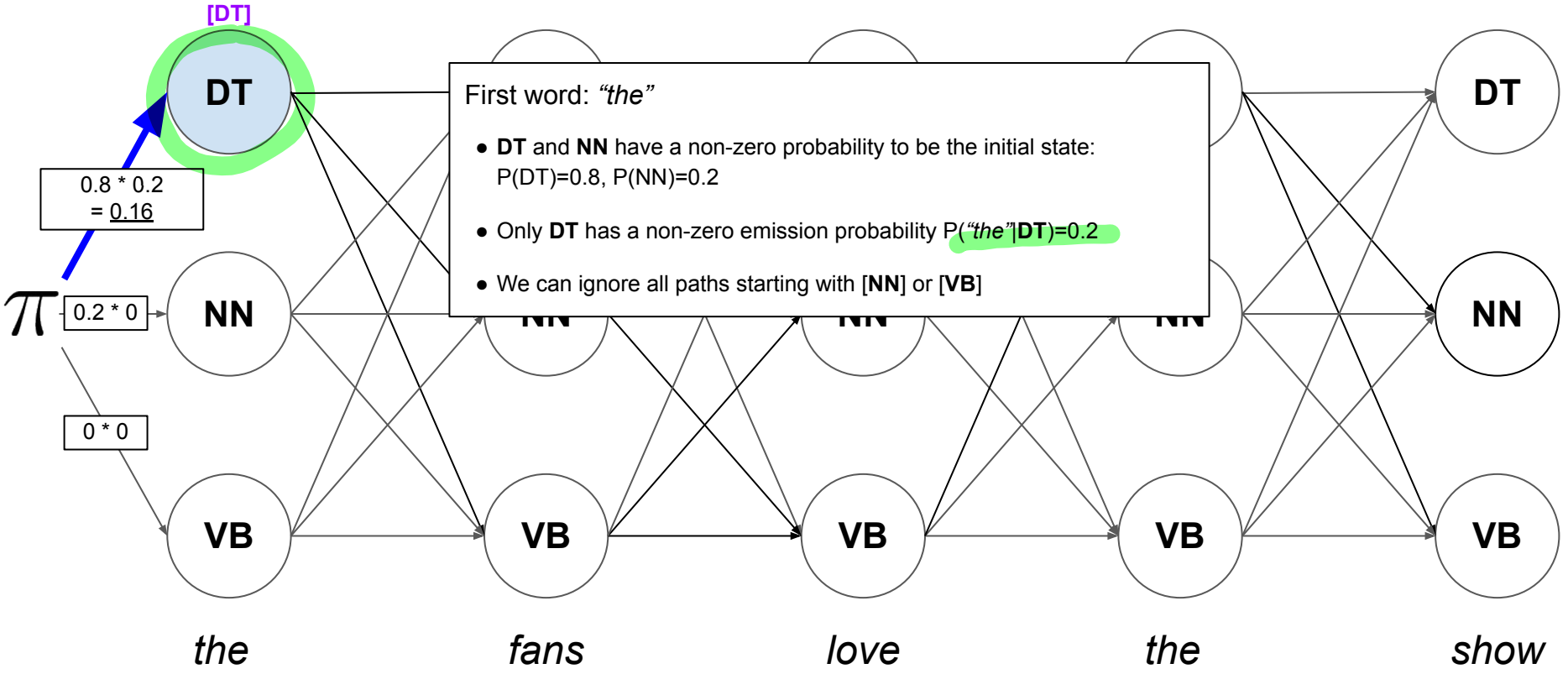


Example

$$\pi = \begin{matrix} \text{DT} & \text{NN} & \text{VB} \\ \end{matrix} \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix}$$

$$A = \begin{matrix} \\ \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \begin{matrix} \begin{matrix} \text{DT} & \text{NN} & \text{VB} \end{matrix} \\ \begin{bmatrix} 0 & 0.8 & 0.2 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix} \end{matrix}$$

$$B = \begin{matrix} \\ \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \begin{matrix} \begin{matrix} \textit{the} & \textit{fans} & \textit{love} & \textit{show} \end{matrix} \\ \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0.0 & 0.05 & 0.3 & 0.1 \\ 0 & 0.25 & 0.15 & 0.3 \end{bmatrix} \end{matrix}$$



First word: *"the"*

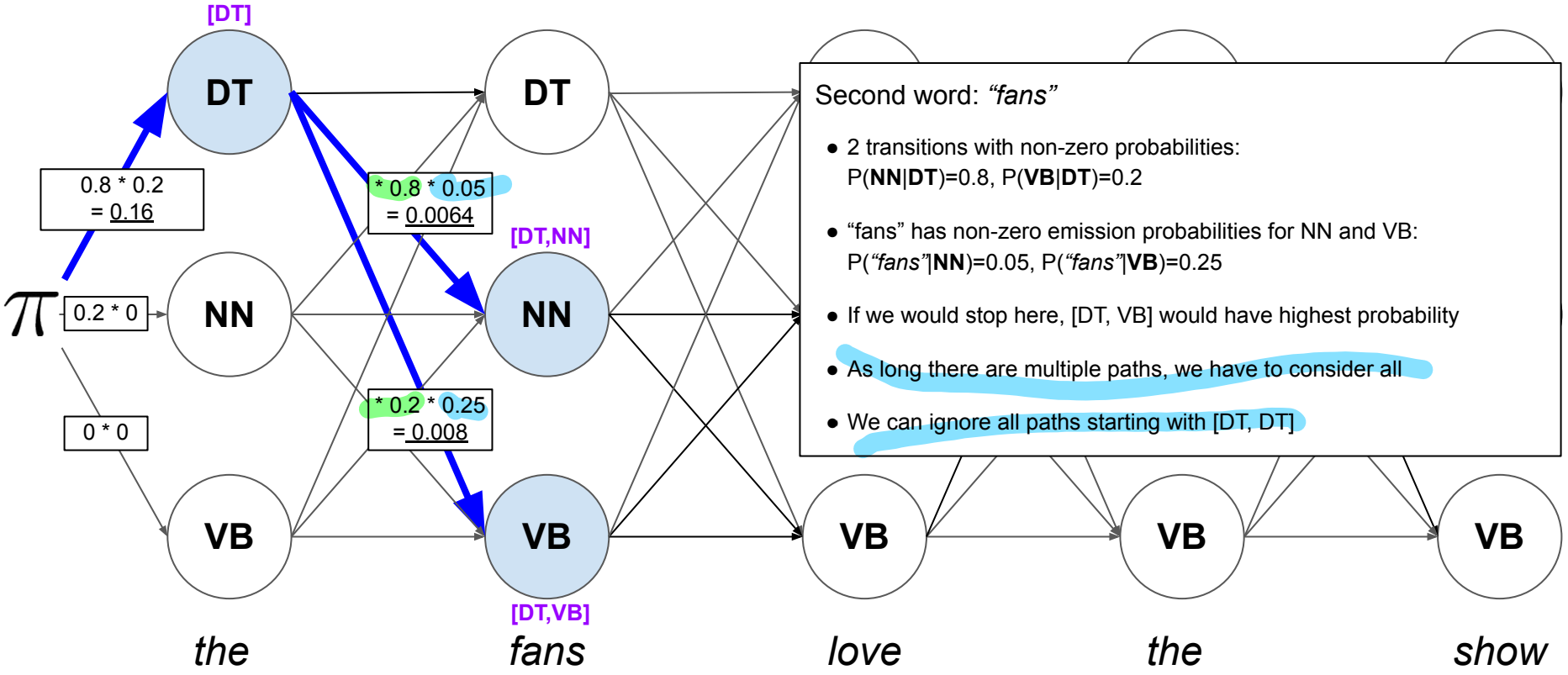
- **DT** and **NN** have a non-zero probability to be the initial state: P(DT)=0.8, P(NN)=0.2
- Only **DT** has a non-zero emission probability P(*"the"*|**DT**)=0.2
- We can ignore all paths starting with [**NN**] or [**VB**]

Example

$$\pi = \begin{matrix} \text{DT} & \text{NN} & \text{VB} \\ \end{matrix} \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix}$$

$$A = \begin{matrix} \\ \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \begin{matrix} \begin{matrix} \text{DT} & \text{NN} & \text{VB} \end{matrix} \\ \begin{bmatrix} 0 & 0.8 & 0.2 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix} \end{matrix}$$

$$B = \begin{matrix} \\ \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \begin{matrix} \begin{matrix} \textit{the} & \textit{fans} & \textit{love} & \textit{show} \end{matrix} \\ \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0.0 & 0.05 & 0.3 & 0.1 \\ 0 & 0.25 & 0.15 & 0.3 \end{bmatrix} \end{matrix}$$



Second word: *"fans"*

- 2 transitions with non-zero probabilities:
  P(**NN**|**DT**)=0.8, P(**VB**|**DT**)=0.2
- "fans" has non-zero emission probabilities for NN and VB:
  P(*"fans"*|**NN**)=0.05, P(*"fans"*|**VB**)=0.25
- If we would stop here, [DT, VB] would have highest probability
- As long there are multiple paths, we have to consider all
- We can ignore all paths starting with [DT, DT]

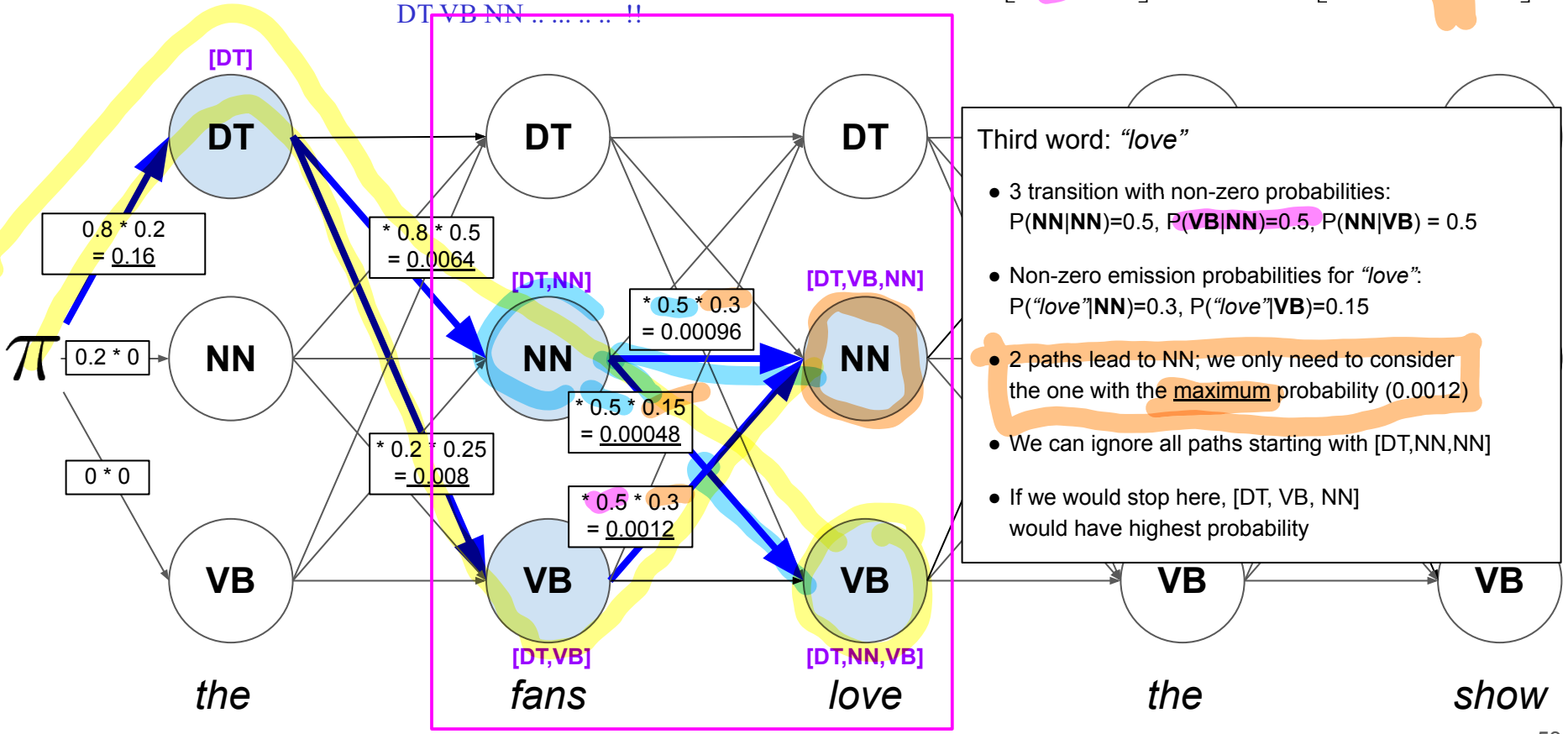
Example

$$\pi = \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix} \quad \text{(DT NN VB)}$$

$$A = \begin{bmatrix} 0 & 0.8 & 0.2 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix} \quad \begin{matrix} \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \quad \text{(columns: DT NN VB)}$$

$$B = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0.0 & 0.05 & 0.3 & 0.1 \\ 0 & 0.25 & 0.15 & 0.3 \end{bmatrix} \quad \begin{matrix} \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \quad \text{(columns: } \textit{the fans love show}\text{)}$$

DT NN NN .. .. !!
DT VB NN .. .. !!



Third word: *"love"*

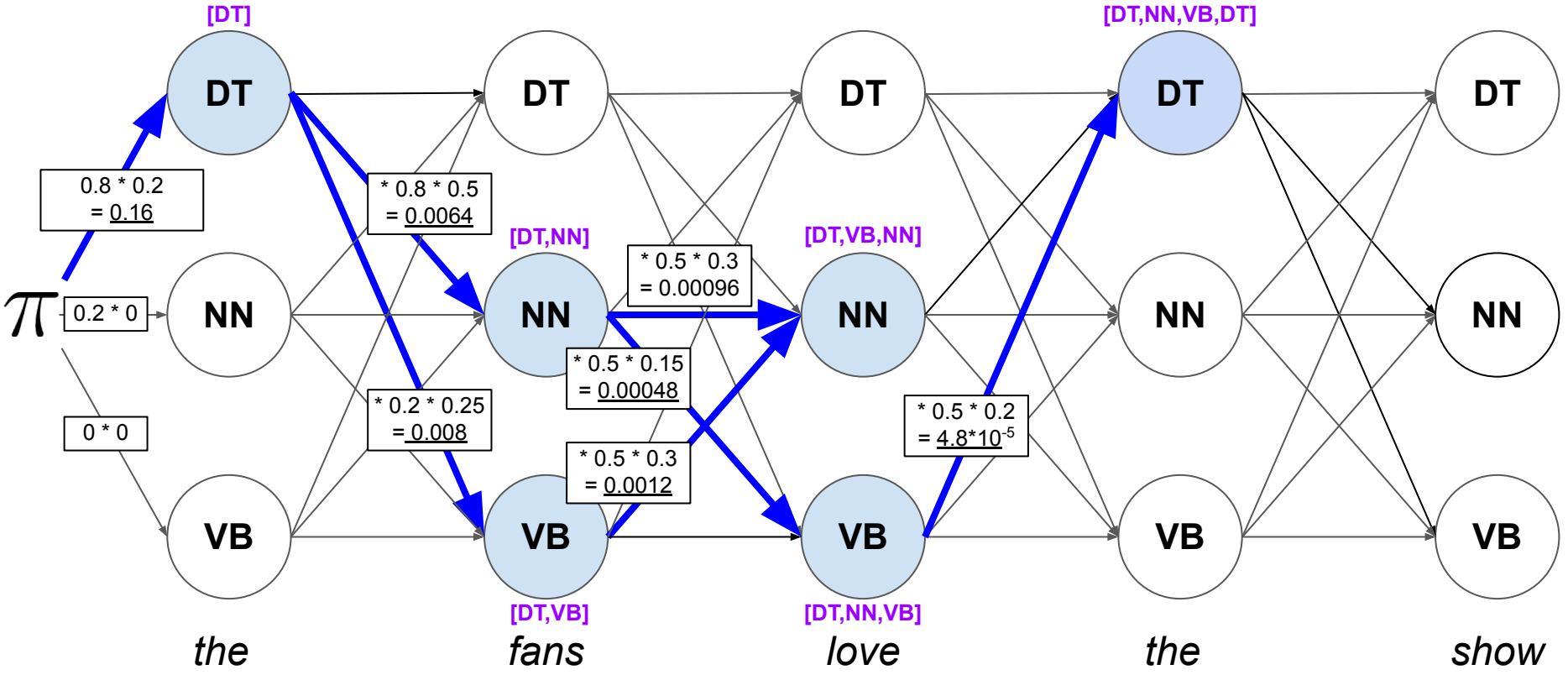
- 3 transition with non-zero probabilities:
  P(**NN**|**NN**)=0.5, P(**VB**|**NN**)=0.5, P(**NN**|**VB**) = 0.5
- Non-zero emission probabilities for *"love"*:
  P(*"love"*|**NN**)=0.3, P(*"love"*|**VB**)=0.15
- 2 paths lead to NN; we only need to consider the one with the maximum probability (0.0012)
- We can ignore all paths starting with [DT,NN,NN]
- If we would stop here, [DT, VB, NN] would have highest probability

Example

$$\pi = \begin{matrix}\text{DT} & \text{NN} & \text{VB}\end{matrix}\begin{bmatrix}0.8 & 0.2 & 0\end{bmatrix}$$

$$A = \begin{matrix}\\ \text{DT}\\ \text{NN}\\ \text{VB}\end{matrix}\begin{matrix}\begin{matrix}\text{DT} & \text{NN} & \text{VB}\end{matrix}\\ \begin{bmatrix}0 & 0.8 & 0.2\\ 0 & 0.5 & 0.5\\ 0.5 & 0.5 & 0\end{bmatrix}\end{matrix}$$

$$B = \begin{matrix}\\ \text{DT}\\ \text{NN}\\ \text{VB}\end{matrix}\begin{matrix}\begin{matrix}\textit{the} & \textit{fans} & \textit{love} & \textit{show}\end{matrix}\\ \begin{bmatrix}0.2 & 0 & 0 & 0\\ 0.0 & 0.05 & 0.3 & 0.1\\ 0 & 0.25 & 0.15 & 0.3\end{bmatrix}\end{matrix}$$

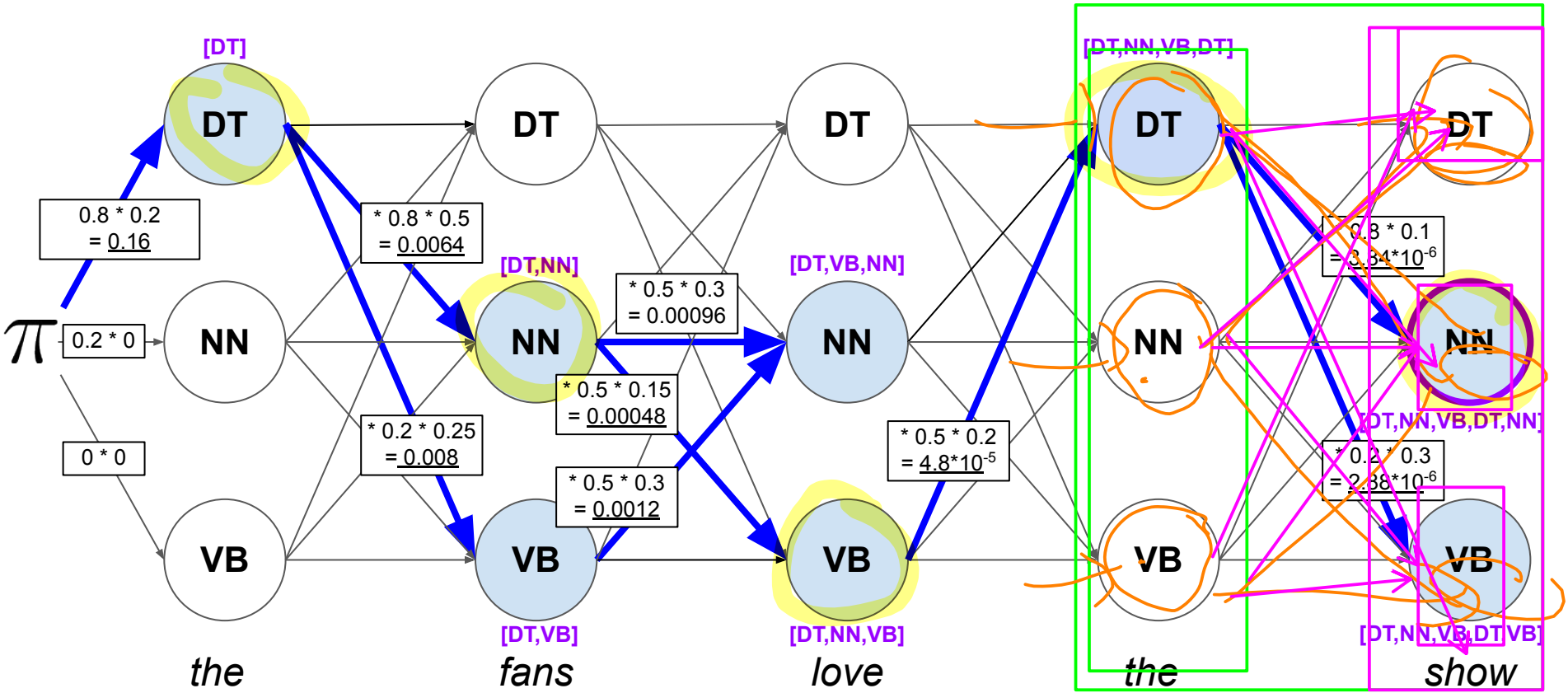


Example

$$\pi = \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix} \quad \begin{matrix} \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix}$$

$$A = \begin{bmatrix} 0 & 0.8 & 0.2 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix} \begin{matrix} \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \quad \begin{matrix} \text{DT} & \text{NN} & \text{VB} \end{matrix}$$

$$B = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0.0 & 0.05 & 0.3 & 0.1 \\ 0 & 0.25 & 0.15 & 0.3 \end{bmatrix} \begin{matrix} \text{DT} \\ \text{NN} \\ \text{VB} \end{matrix} \quad \begin{matrix} \textit{the} & \textit{fans} & \textit{love} & \textit{show} \end{matrix}$$

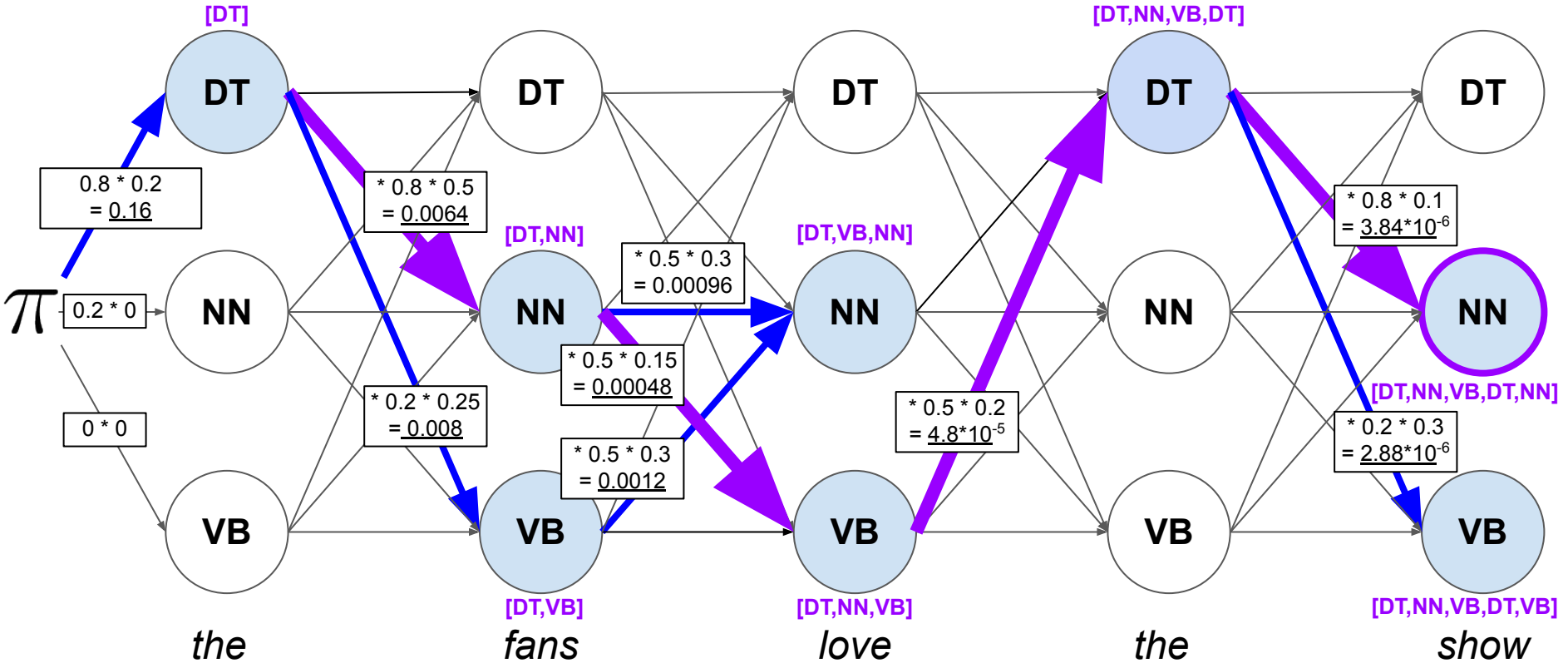


Viterbi Algorithm

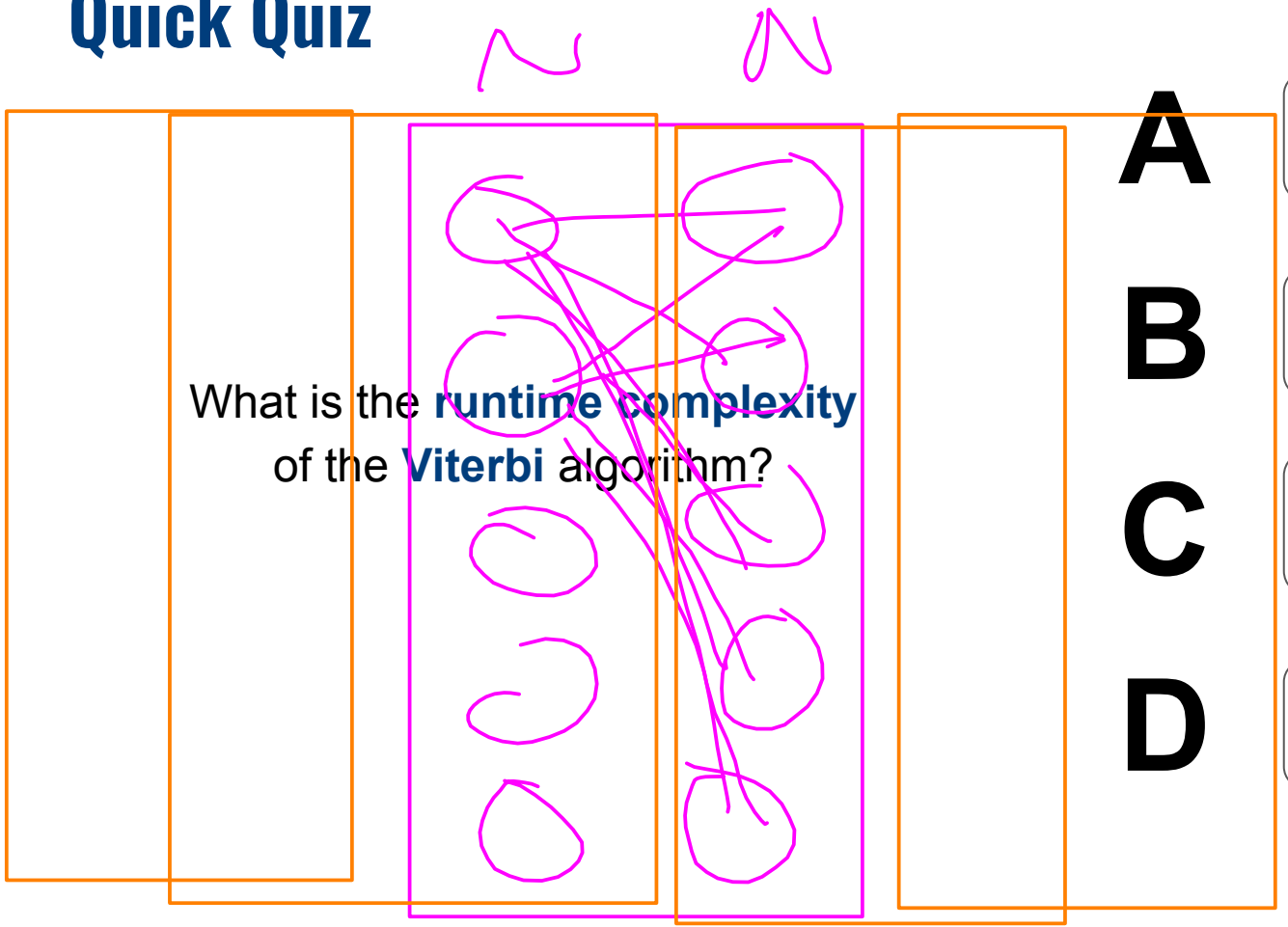
- 2 important questions
  - How to get the final state sequence with the highest probability?
  - How exactly does the Viterbi algorithm reduces complexity?

Backtracking

- During forward pass: remember input path with max probability
- Backtracking: follow paths with max probabilities back to beginning



Quick Quiz



What is the **runtime complexity** of the **Viterbi** algorithm?

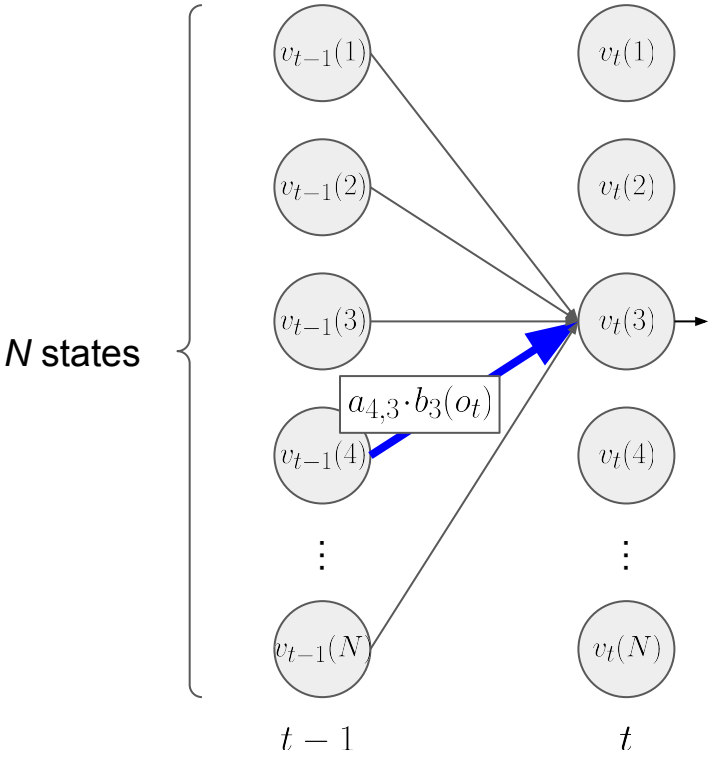
A $O(N + T)$

B $O(N \cdot T)$

C $O(N^2 \cdot T)$

D $O(N^2 \cdot T^2)$

Viterbi Algorithm — Complexity Analysis



- Let $v_t(3)$ be maximal for the path coming from $v_{t-1}(4)$
- We can ignore all paths coming from $v_{t-1}(j), \quad j \neq 4$
- This holds true for all steps t and states j

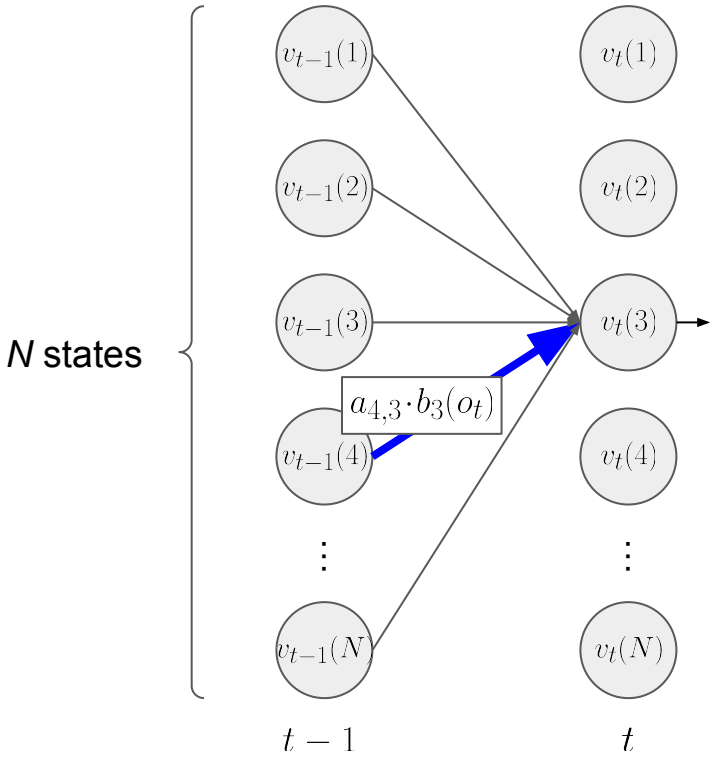
➜ Time complexity for Viterbi: $O(T \cdot N^2)$

length of sequence

states

Note: Cases where $a_{ij} \cdot b_j(o_t) = 0$ might be an additional convenience but not the main reason for the polynomial complexity

Viterbi Algorithm — The Basic Algorithm



Initialization

$$v_1(t) = \pi_j \cdot b_j(o_1) \qquad 1 \le j \le N$$

$$bt_1(t) = 0$$

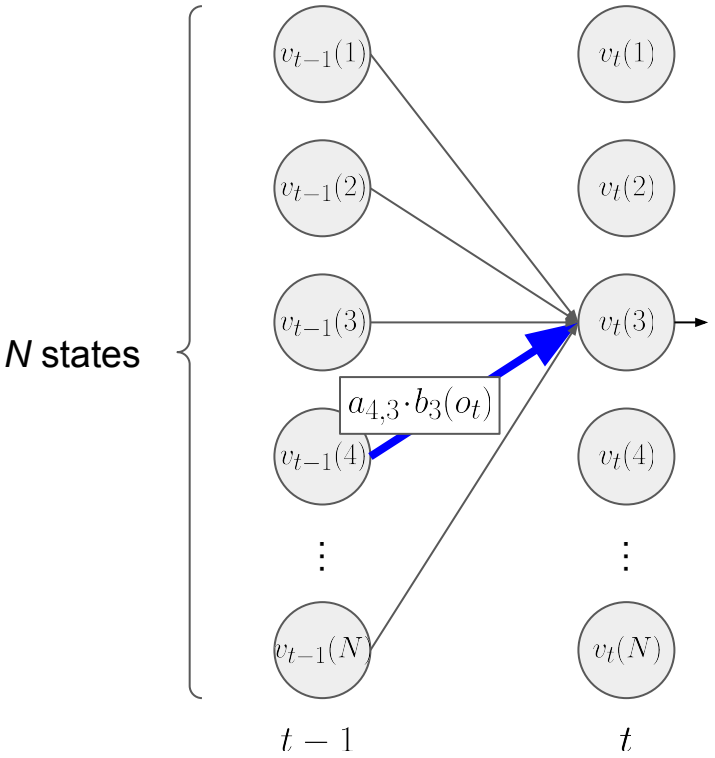
Recursion

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

$$bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

Example for backtrace: $bt_t(3) = 4$ since we get the highest probability for $v_t(3)$ from the path coming from $v_{t-1}(4)$

Viterbi Algorithm — The Basic Algorithm



Termination (after computing all $v_t(j)$ and $bt_t(j)$)

Probability of most likely path: $P^* = \max_{i=1}^{N} v_T(i)$

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

Viterbi Algorithm — Practical Consideration

- The “usual” problem: Risk of arithmetic underflow

$$v_1(t) = \pi_j \cdot b_j(o_1)$$

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

Values for $v_t(j)$ become very small as we multiple many (potentially very) small probability values

➜ The “usual” solution: Logarithm

$$v_1(t) = \log \pi_j + \log b_j(o_1)$$

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

Viterbi Algorithm — Python/NumPy Implementation

```python
def viterbi(tokens, A, B, PI):
    N, T = A.shape[0], len(tokens)
    M = np.zeros((N, T))                    # Reflecting probabilties of trellis
    BT = np.zeros((N, T), dtype=np.int16)   # For the Backtracking pointers

    # Initialization
    for s in range(N):
        M[s,0] = PI[s] * B[s, word2index[tokens[0]]]

    # Recursion (with dynamic programming)
    for t in range(1, T):
        for s in range(N):
            new_probs = M[:,t-1] * A[:,s] * B[s, word2index[tokens[t]]]
            max_idx = np.argmax(new_probs)
            M[s,t]  = new_probs[max_idx]
            BT[s,t] = max_idx

    # Termination (start backtracking)
    state = np.argmax(M[:,-1])
    state_sequence = []
    for i in reversed(range(T)):
        state_sequence.append(state)
        state = BT[:,i][state]

    return [ index2tag[idx] for idx in reversed(state_sequence) ]
```

Note: This slide is only to show that it does not take much code to implement the Viterbi algorithm.

$$v_1(t) = \pi_j \cdot b_j(o_1) \qquad bt_1(t) = 0$$

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

$$bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

$$q_T^* = \operatorname{argmax}_{i=1}^{N} v_T(i)$$

Viterbi Algorithm — Python/NumPy Implementation

- Using the HMM trained over 25k movie reviews
  - 50 states (POS tags)
  - 83k+ tokens (words, punctuation marks, etc.)

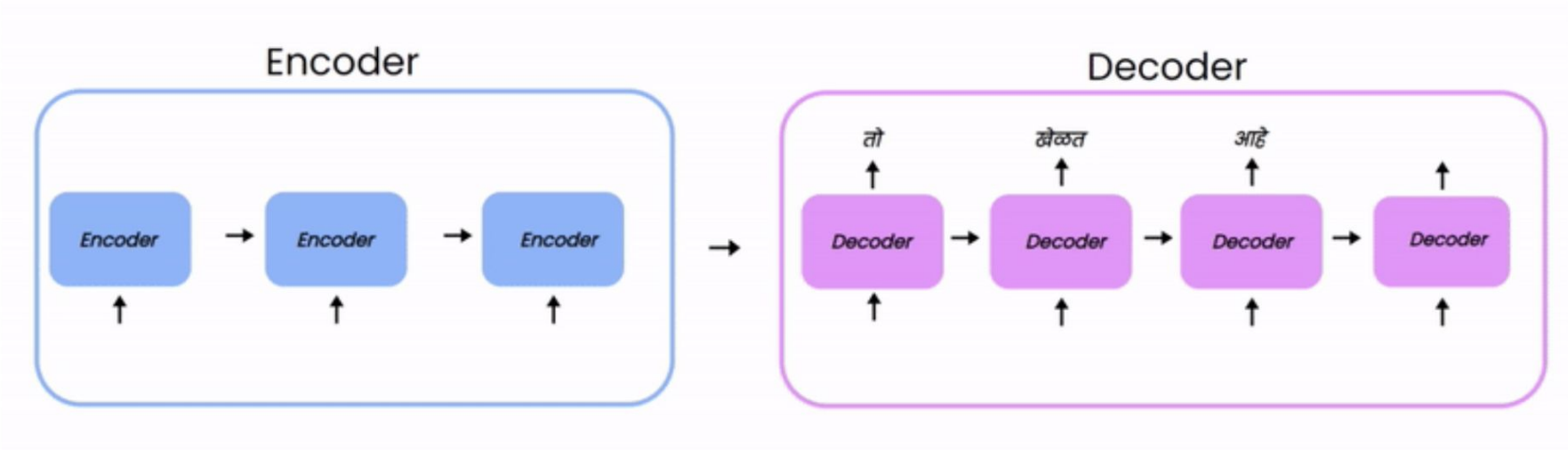
Important: we've cheated here by annotating the reviews using spaCy, not humans!

```
viterbi(['the', 'fans', 'love', 'the', 'show'], A, B, PI)
['DT', 'NNS', 'VBP', 'DT', 'NN']

viterbi(['the', 'fans', 'like', 'the', 'show'], A, B, PI)
['DT', 'NNS', 'IN', 'DT', 'NN']

viterbi(['funny', 'movies', 'are', 'the', 'best'], A, B, PI)
['JJ', 'NNS', 'VBP', 'DT', 'JJS']

viterbi(['i', 'like', 'watching', 'comedies'], A, B, PI)
['PRP', 'VBP', 'VBG', 'NNS']
```

Summary

- Sequences
  - A primary form of natural language data with many applications
    (a sentence is sequence of words; sequence captures meaning ➜ BoW model intrinsically limited)
  - Many sequence tasks in NLP
- Focus of this lecture: sequence labeling
  - POS tagging as very fundamental sequence labeling task
  - Different approaches, incl. Hidden Markov Models (HMM)
- Next lecture: encoder–decoder architecture
  - Neural network-based architecture
  - Applicable to all sequence tasks

Outlook for Next Week: Encoder–Decoder



GIF Credits Aditya Shirsath @ Medium

Pre-Lecture Activity for Next Week

- Assigned Task
  - Post a 1-2 sentence answer to the following question in the [Pre-Lecture] discussion

“What is an encoder? What is a decoder?
What are we trying to encode/decode anyways?”

Side notes:
- This task is meant as a warm-up to provide some context for the next lecture
- No worries if you get lost; we will talk about this in the next lecture
- You can just copy-&-paste others' answers but this won't help you learn better