CS639 Deep Learning for NLP

Sequence Labeling I

Junjie Hu

Slides adapted from Luke, Yulia, Bob

https://junjiehu.github.io/cs639-spring26/

Outline

- Sequence Labeling
- (**Generative**) **Hidden Markov Model**
- (**Discriminative**) **Conditional Random Field** (next lecture)

Sequence labeling problems

- Map **a sequence of words** to **a sequence of labels**
  - Part-of-speech tagging (Church, 1988; Brants, 2000)
  - Named entity recognition (Bikel et al., 1990)
  - Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
  - Word alignment of parallel text (Vogel et al., 1996)
  - Compression (Conroy and O'Leary, 2001)
  - Acoustic models, discourse segmentation, etc.

Syntax: Part-of-Speech tagging

- **Open classes** allow new members through borrowing (e.g., the noun *cafe*) and derivation (e.g., the adjective *bounteous* from the noun *bounty*). A word can have new POS tag memberships.
  - Nouns
  - Verbs
  - Adjectives
  - Adverbs

- **Closed classes** of words do not allow new members and usually involve grammatical rather than lexical words. Namely, each word has a closed, fixed POS tag membership. Reasonably easy to enumerate.
  - Prepositions
  - Determiners
  - Pronouns
  - Conjunctions
  - Auxiliary verbs

Part of Speech Tagging

- Penn treebank tagset (Marcus et al., 1993)

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

POS tagging (Example)

- System outputs:
  - The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
  - There/EX are/VBP 70/CD children/NNS there/RB
  - Preliminary/JJ findings/NNS were/VBD reported/VBN in/IN today/NN 's/POS New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

Universal Dependencies for All Languages

Universal Dependencies

Universal Dependencies (UD) is a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. UD is an open community effort with over 300 contributors producing more than 150 treebanks in 90 languages. If you're new to UD, you should start by reading the first part of the Short Introduction and then browsing the annotation guidelines.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Why POS tagging?

- Goal: resolve ambiguities

- Text-to-speech
  - Words w/ slightly different pronunciations denoting different POS, e.g., record/N →/ˈrekərd/, record/V → /rəˈkôrd/

- Lemmatization
  - saw/V → see, saw/N → saw

- Preprocessing for harder disambiguation problems
  - Syntactic parsing
  - Semantic parsing

Sequence labeling as text classification

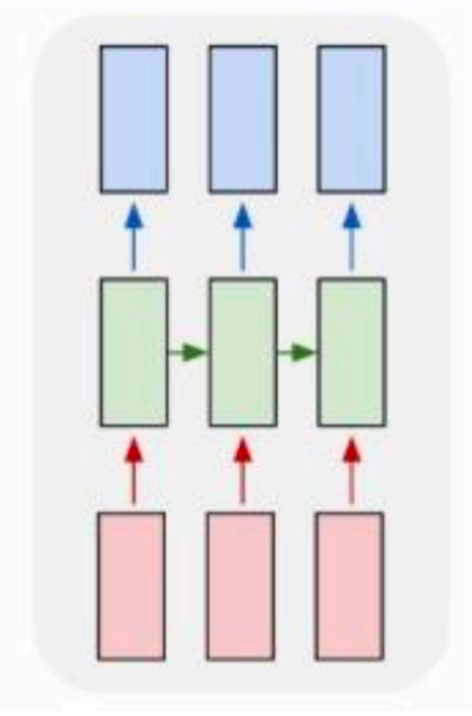
- **Generative** Model: Learn joint probability $P(X, Y)$
  - Hidden Markov Models

$$\hat{Y} = \arg \max_{\substack{y_1 \cdots y_n \\ \forall y_i \in \mathcal{C}}} P(x_1 \cdots x_n, y_1 \cdots y_n)$$

- **Discriminative** Model: Learn conditional probability $P(Y|X)$
  - Conditional Random Fields
  - Neural network-based methods

$$\hat{Y} = \arg \max_{Y} P(Y|X)$$

- Both trained via Maximum Likelihood Estimation

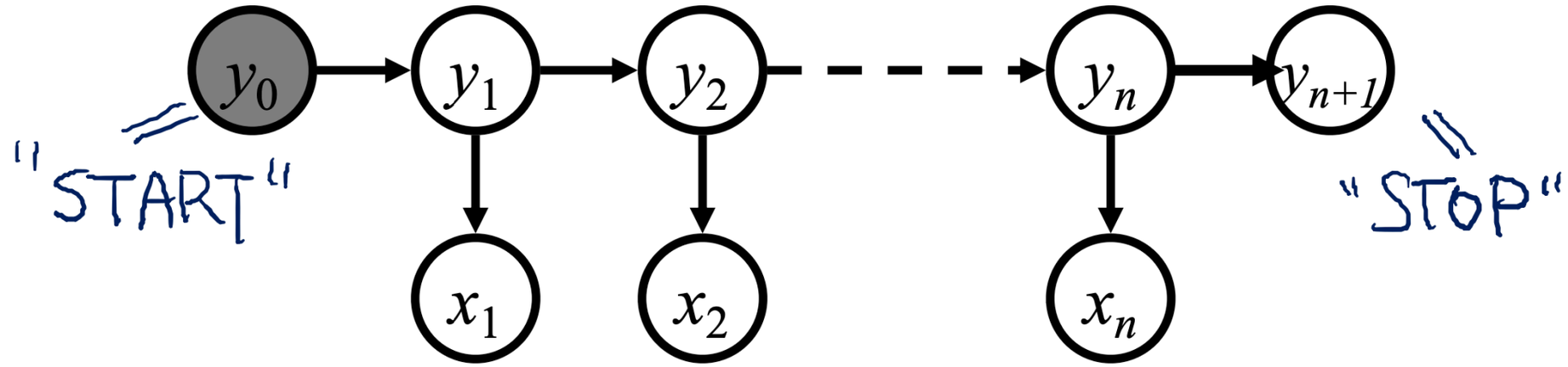


Hidden Markov Model

(Sequential Version of Naive Bayes)

Classic Solution: HMMs

- We want a model of unobservable (hidden) sequences y and observations x



$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = q(\text{STOP}|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

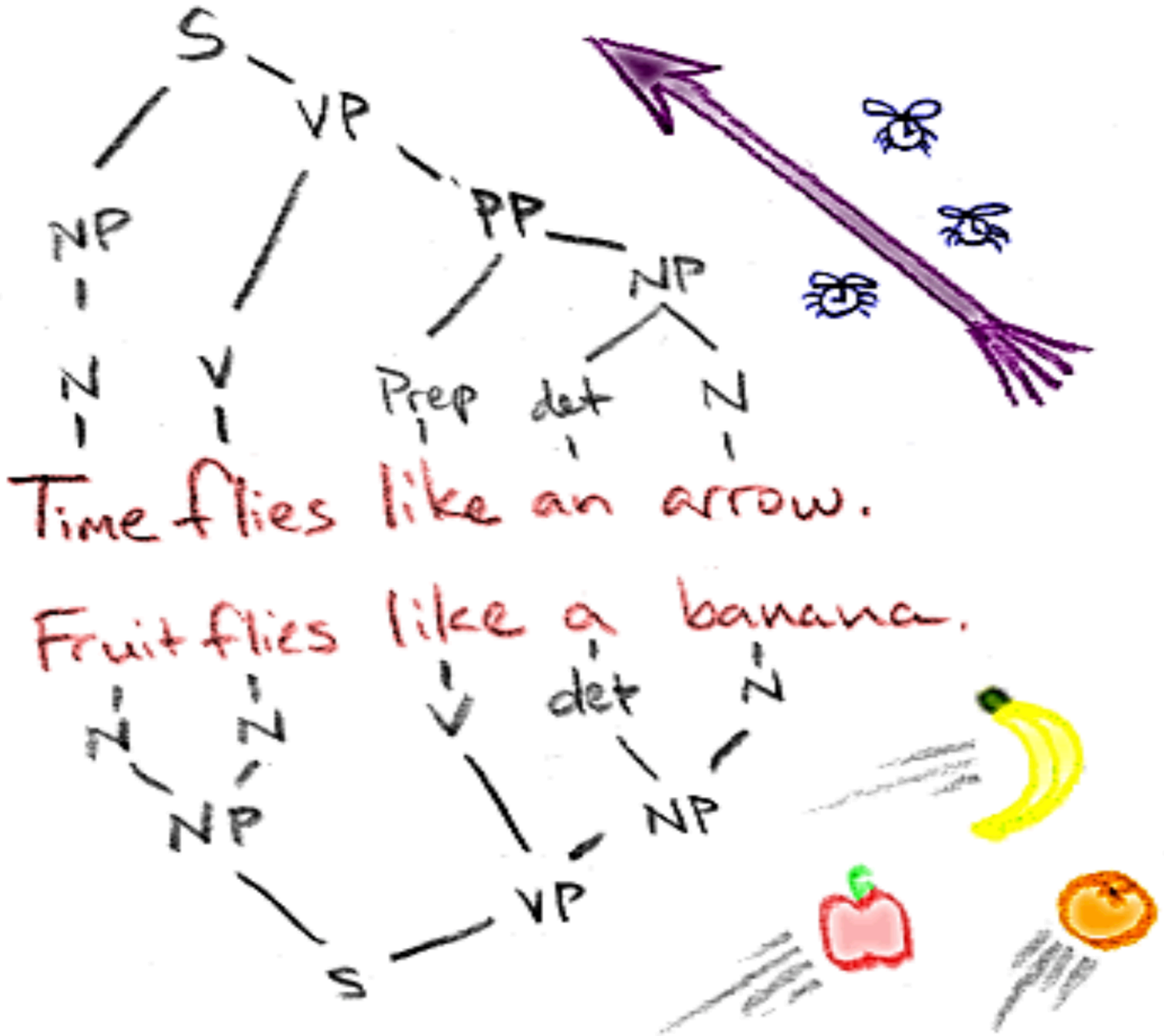
where $y_0 = \text{START}$ and we call $q(y'|y)$ the transition distribution and $e(x|y)$ the emission (or observation) distribution.

Assumptions:

- Tag/state sequence is generated by a Markov model
- Words are chosen independently, conditioned only on the tag/state
- These are totally broken assumptions: why?

Tag predictions depends on context

- Time flies like an arrow
- Fruit flies like a banana



HMM Learning and Inference

- **Learning** by **maximum likelihood estimation**: transition $q(y'|y)$ and emissions $e(x|y)$

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = q(\text{STOP}|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

- **Inference** (linear time in sentence length!)

  - Viterbi:

$$y* = \underset{y_1 \ldots y_n}{\text{argmax}}\ p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) \quad \text{where } y_{n+1} = \text{STOP}$$

  - Forward Backward:

$$p(x_1 \ldots x_n, y_i) = \sum_{y_1 \ldots y_{i-1}} \sum_{y_{i+1} \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

Learning: Maximum Likelihood

- Supervised Learning
  - Assume m fully labeled training examples:

$$\{(x^{(i)}, y^{(i)})|i = 1 \cdots m\}$$

where $x^{(i)} = x_1 \cdots x_n$ and $y^{(i)} = y_1 \cdots y_n$

- What's the maximum likelihood estimate?

$$p(x_1 ... x_n, y_1 ... y_{n+1}) = q(\text{STOP}|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

$q_{ML}(y_i|y_{i-1})$ $e_{ML}(x_i|y_i)$

Learning: Maximum Likelihood

- MLE: counting the co-occurrence of the event

$$q_{ML}(y_i|y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \qquad e_{ML}(x|y) = \frac{c(y, x)}{c(y)}$$

- Will these estimates be high quality?
  - Which is likely to be more sparse, *q* or *e*?
    - The emission function, because $c(y, x)$ is more likely to have sparse values.

- Can use all the same smoothing tricks we used for counting-based language models!

- Other approaches: Map low-frequency words to a small, finite set of units (e.g., prefixes, word classes), and run MLE on new sequences

Inference (Decoding)

- Problem: find the most likely (Viterbi) sequence under the model

$$y* = \underset{y_1 \ldots y_n}{\text{argmax}}\ p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

- Given model parameters, we can score any sequence pair

NNP	VBZ	NN	NNS	CD	NN	.
Fed	raises	interest	rates	0.5	percent	.

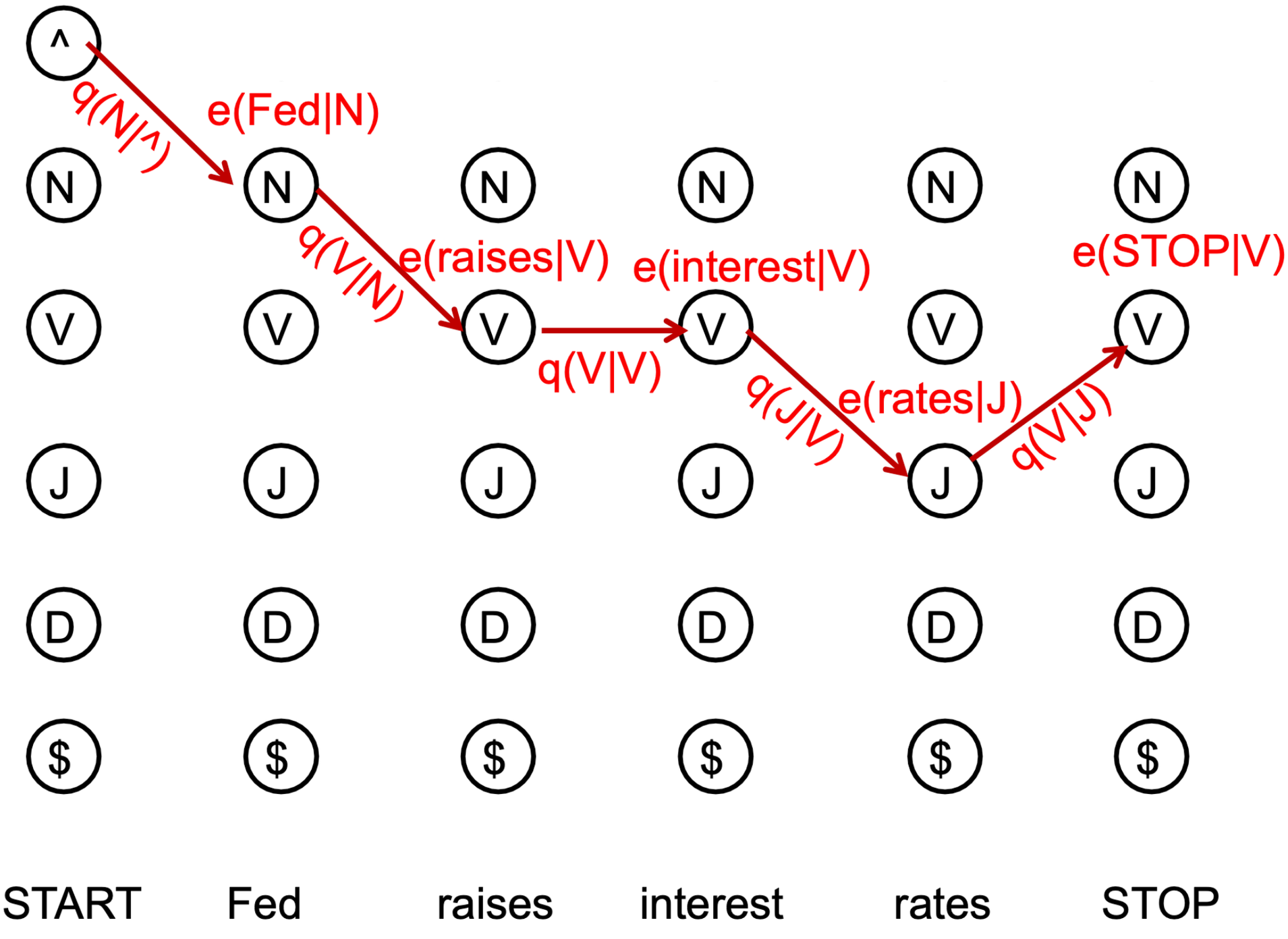
- In principle, we can list all possible tag sequences, score each one, and pick **the best one (a.k.a. the Viterbi state sequence)**

NNP VBZ NN NNS CD NN ⇨ logP = -23

NNP NNS NN NNS CD NN ⇨ logP = -29

NNP VBZ VB NNS CD NN ⇨ logP = -27

The State Lattice/Trellis: Viterbi



- Brute force approach: enumerate $n^{\mathcal{K}}$ possible tag sequences

Dynamic Programming!

- Focus on max, consider special case of n=2

- Define $\pi(i, y_i)$ to be the max score of a sequence of length i ending in tag y_i

$$\max_{y_1,y_2} q(STOP|y_2)q(y_2|y_1)e(x_2|y_2)q(y_1|START)e(x_1|y_1)$$

$$= \max_{y_2} q(STOP|y_2)e(x_2|y_2) \max_{y_1} q(y_1|START)q(y_2|y_1)e(x_1|y_1)$$

$$= \max_{y_2} q(STOP|y_2)e(x_2|y_2)\pi(2, y_2)$$

given that $\pi(2, y_2) = \max_{y_1} q(y_1|START)q(y_2|y_1)e(x_1|y_1)$

- What about the general case? (Consider n=3, etc…)

Dynamic Programming!

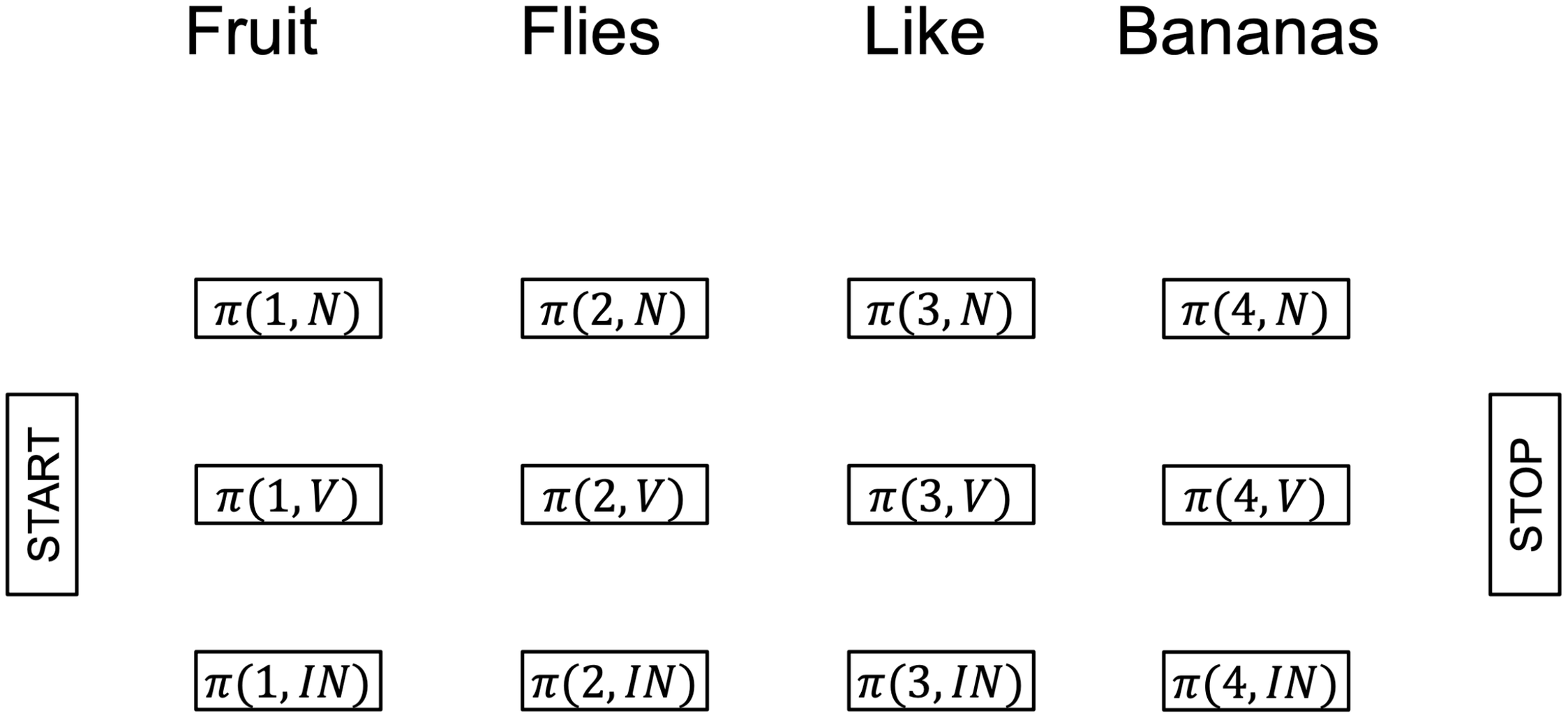
- General case

- Define $\pi(i, y_i)$ to be the max score of a sequence of length i ending in tag y_i

$$\begin{aligned}
\pi(i, y_i) &= \max_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i) \\
&= \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \max_{y_1 \dots y_{i-2}} p(x_1 \dots x_{i-1}, y_1 \dots y_{i-1}) \\
&= \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \quad \pi(i-1, y_{i-1})
\end{aligned}$$

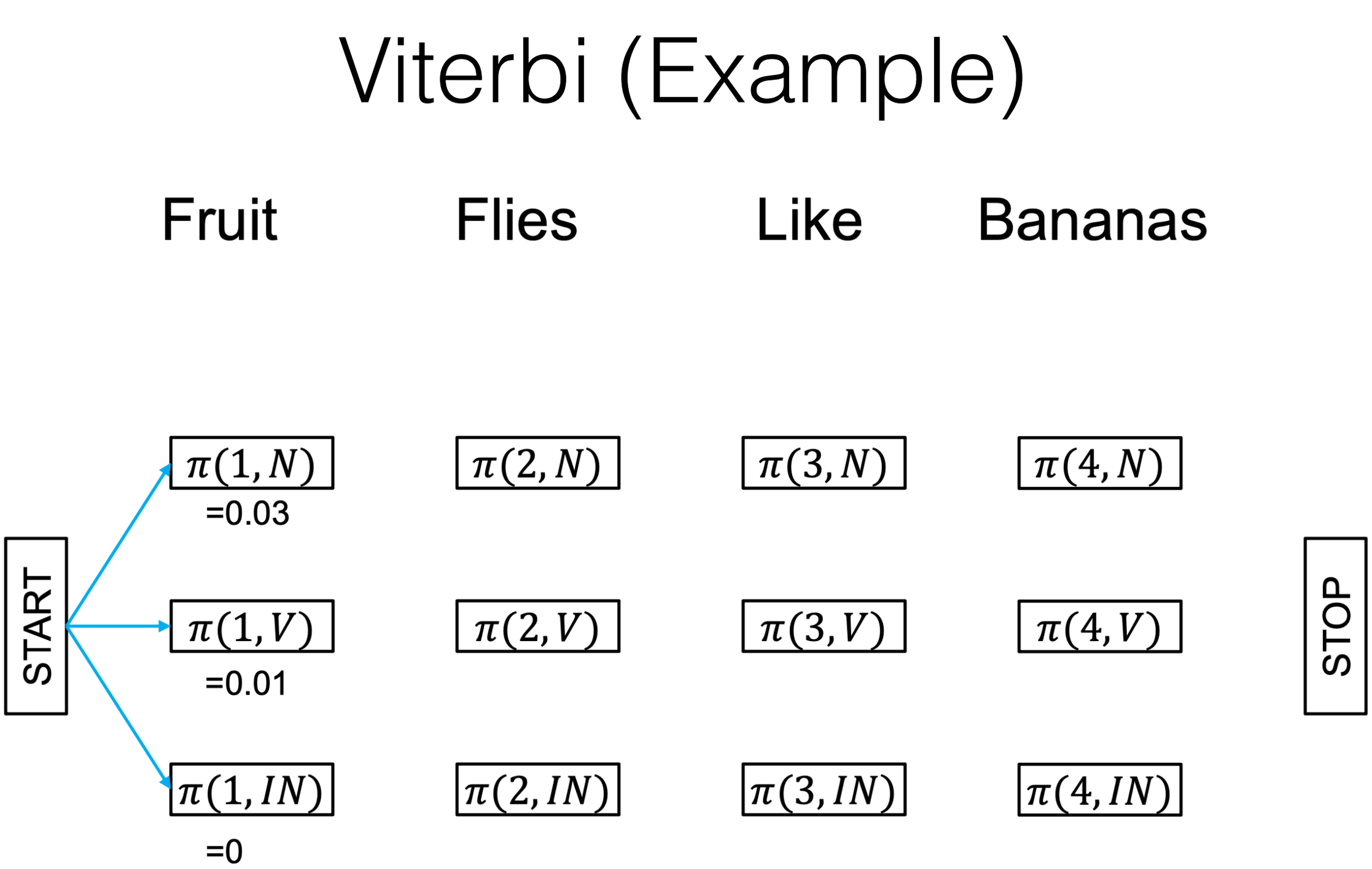
- We now have an efficient algorithm. Start with *i=0* and work your way to the end of the sentence!

Viterbi (Example)



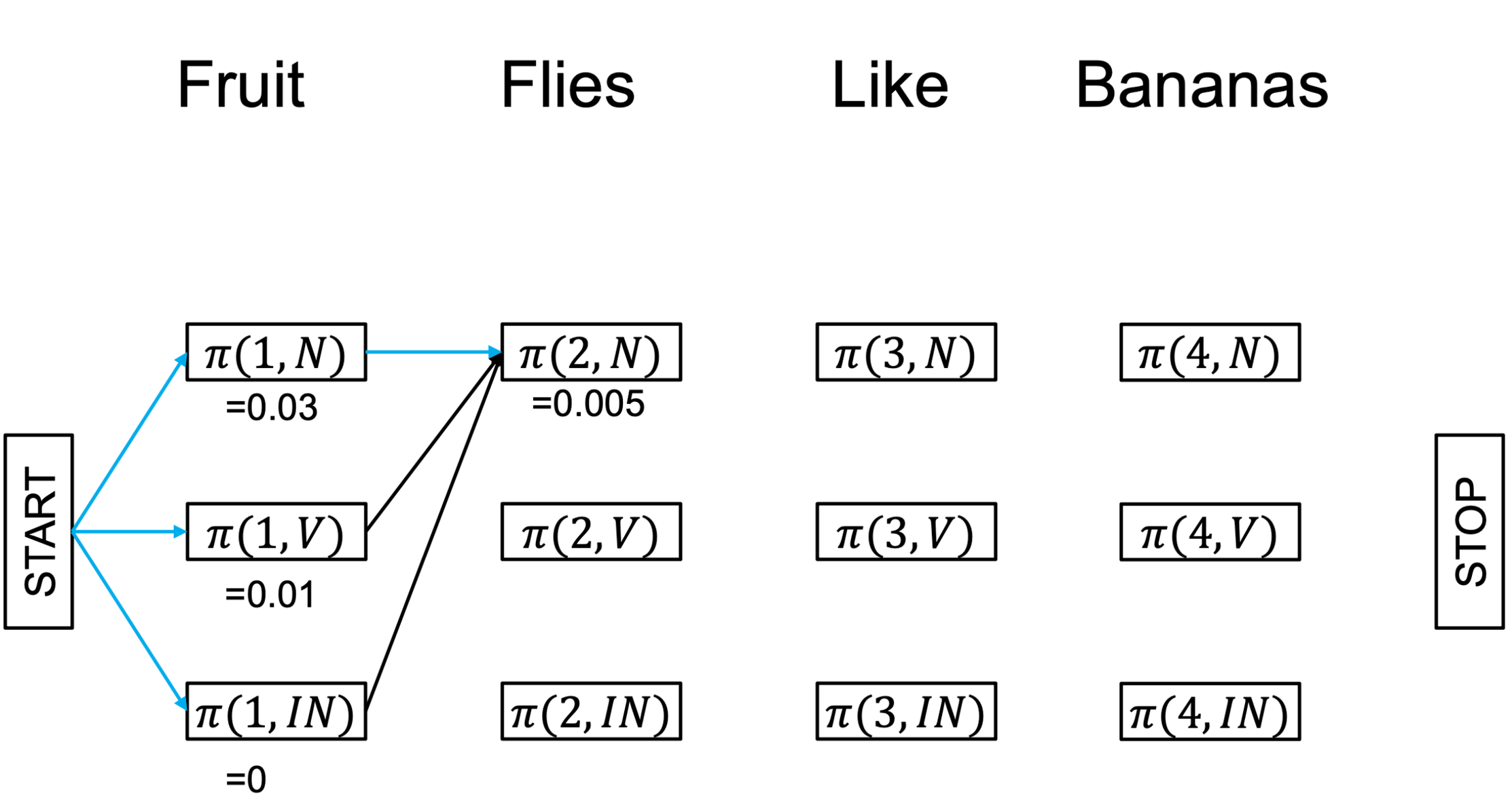
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

Viterbi (Example)



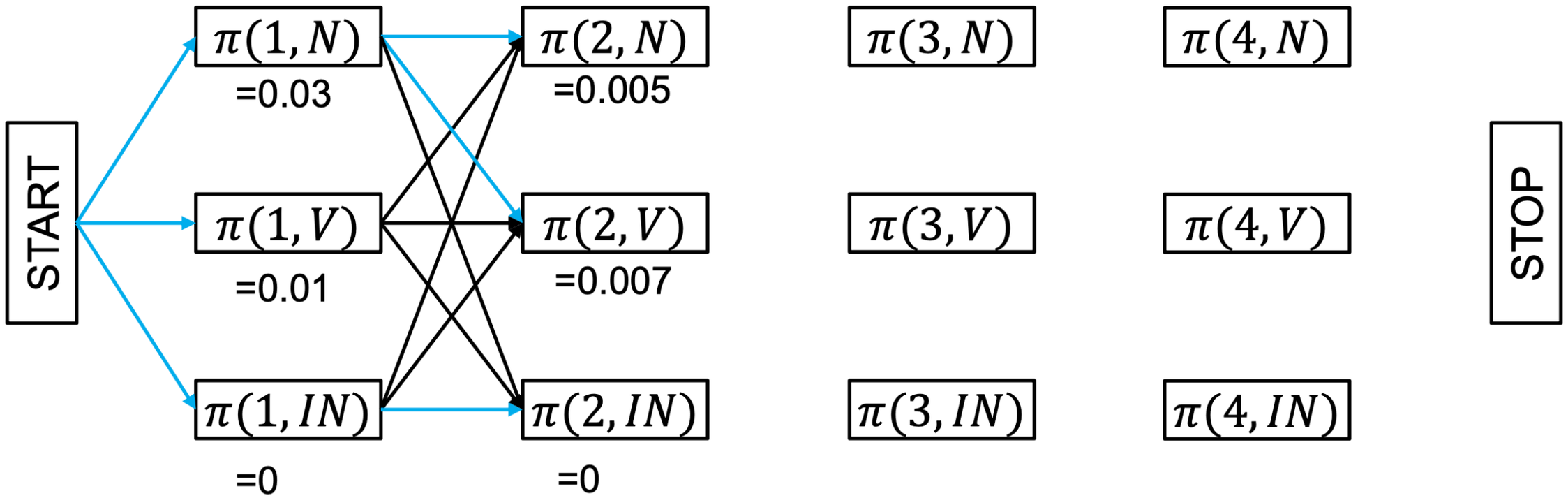
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

Viterbi (Example)



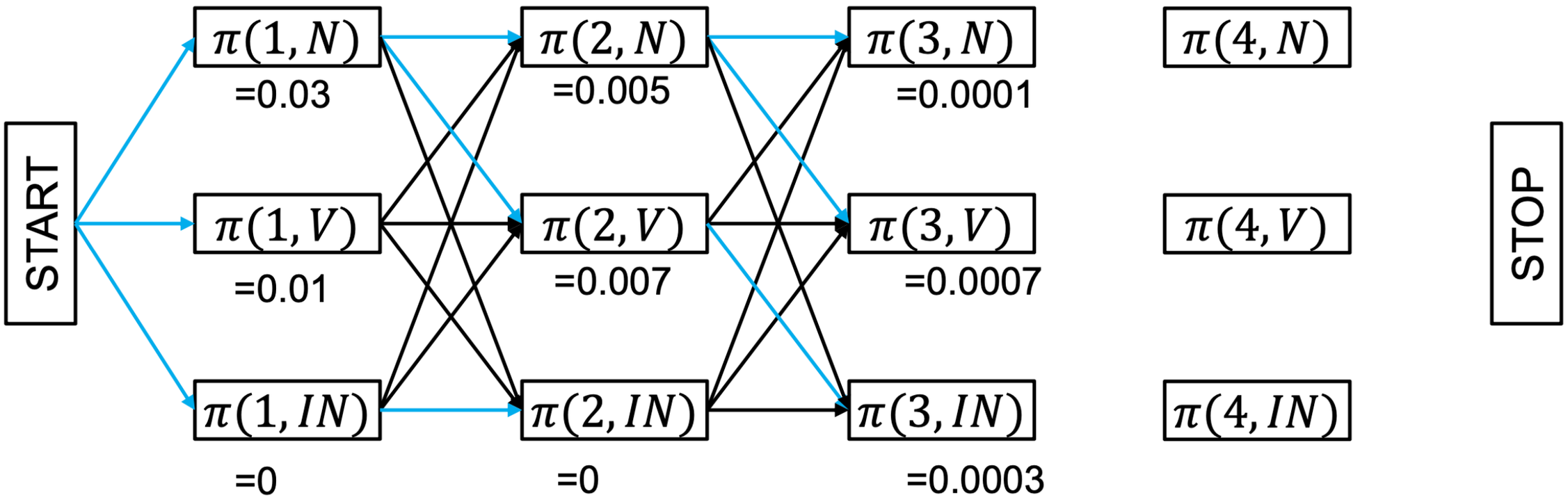
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

Viterbi (Example)



$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

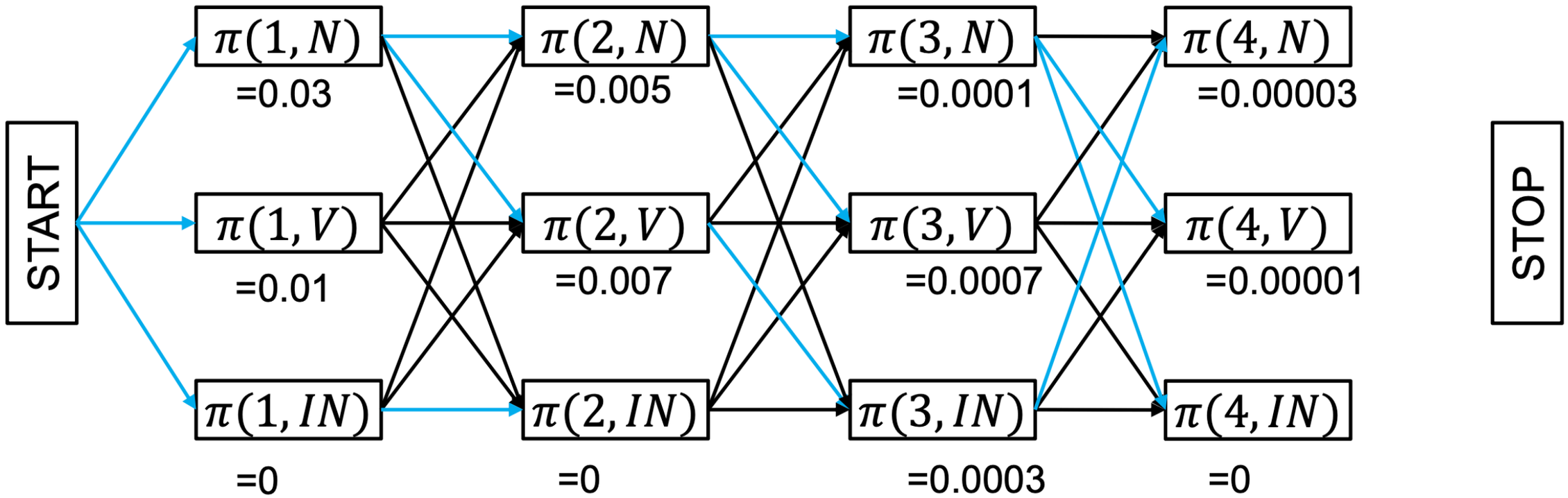
Viterbi (Example)



$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

Viterbi (Example)

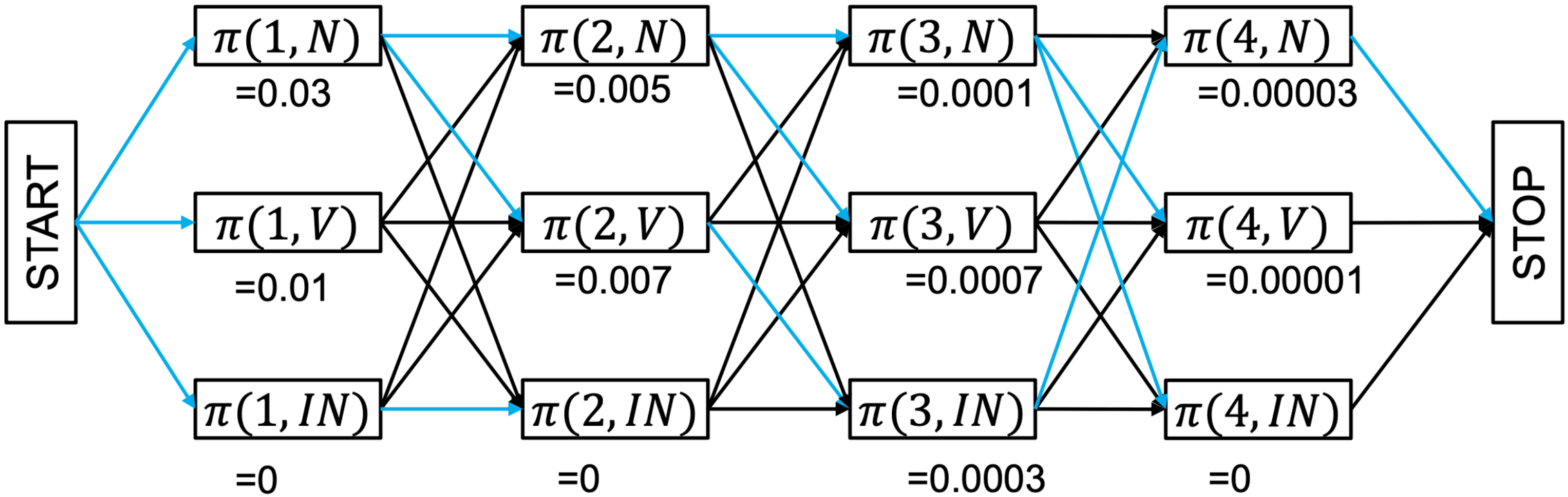
Fruit Flies Like Bananas



$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

Viterbi (Example)

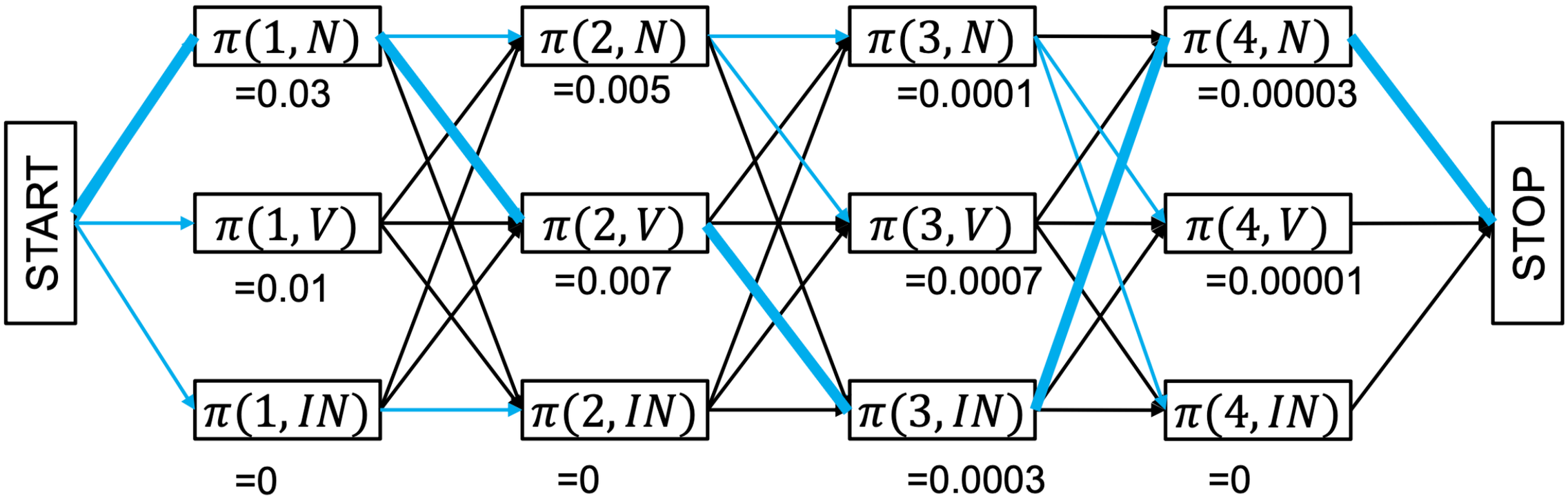
Fruit Flies Like Bananas



$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

Viterbi (Example)

Fruit Flies Like Bananas



$$bp(i, y_i) = \arg\max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

Why is this not a greedy algorithm? Why does this find max P(.)?

Viterbi Algorithm

- Dynamic programming (for all i)

$$\pi(i, y_i) = \max_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i)$$

  - Iterative computation

$$\pi(0, y_0) = \begin{cases} 1 \text{ if } y_0 == START \\ 0 \text{ otherwise} \end{cases}$$

For i = 1 … n:

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

  - Store back pointers:

$$bp(i, y_i) = \arg\max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

- What is the final solution? $bp(n+1, STOP)$

Viterbi Algorithm: Time complexity

- Linear in sentence length *n*

- Polynomial in the number of possible tags $\mathcal{K}$

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

iterate over all possible tags

- Specifically:

$$O(n|\mathcal{K}|) \text{ entries in } \pi(i, y_i)$$

$$O(|\mathcal{K}|) \text{ time to compute each } \pi(i, y_i)$$

- Total runtime:

$$O(n|\mathcal{K}|^2)$$

Questions?