CS769 Advanced NLP

Morphology & Sequence Labeling I

Junjie Hu

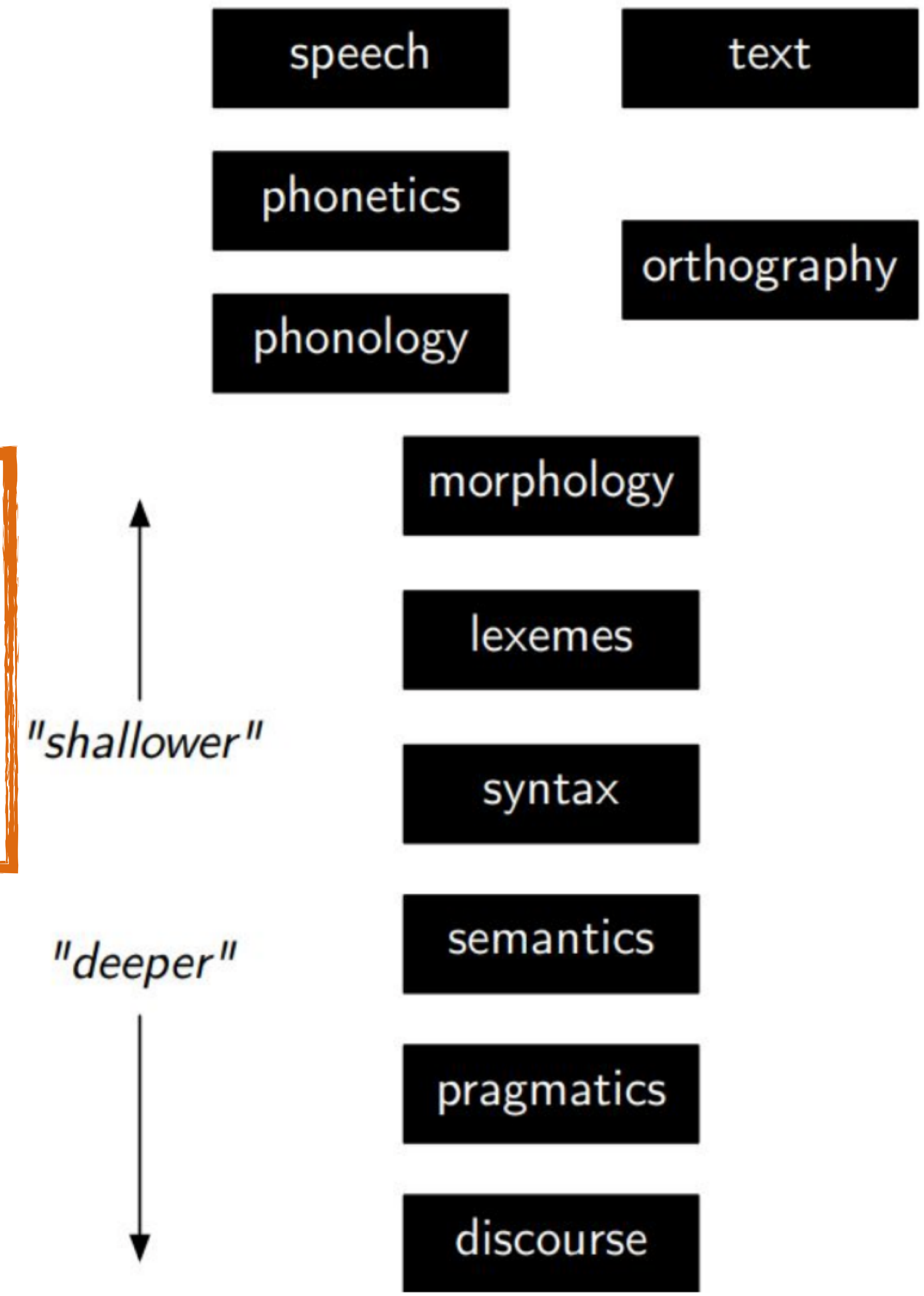
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Luke, Yulia, Bob

https://junjiehu.github.io/cs769-spring22/

Levels of Linguistic Knowledge

Phonetics	The study of the sounds of human language
Phonology	The study of sound systems in human language
Morphology	The study of the formation and internal structure of words
Syntax	The study of the formation and internal structure of sentences
Semantics	The study of the meaning of sentences
Pragmatics	The study of the way sentences with their semantic meanings are used for particular communicative goals



Morphology
& Word Tokenization

Morphology: Internal Structure of Words

- **Derivational morphology**: How new words are created from existing words
    - [grace]
    - [[grace]ful]
    - [un[[grace]ful]]

- **Inflectional morphology**: How features relevant to the syntactic context of a word are marked on that word.
    - This student walks.
    - These students walk.
    - These students walked.

- **Compounding**: Creating new words by combining existing words.
    - With or without space: surfboard, golf ball, blackboard

Morphemes

- **Morphemes**. Minimal pairings of form and meaning.
  - **Roots**: the “core” of a word that carries its basic meaning
    - E.g., apple, walk.
  - **Affixes** (prefixes, suffixes, infixes, and circumfixes). Morphemes that are added to a root (or a stem) to perform either derivational or inflectional functions.
    - Prefix: *un-* → negation
    - Suffix: *-s* → plural noun
    - Infix: *-ít-* → Spanish name adapted from English, e.g., Victor → Victítor
    - Circumfix: *ge- … -t* → German past participle

Morphological Parsing

- **Input**: a word
- **Output**: the word's stem(s) and features expressed by other morphemes.

Example:

- geese → goose + N +Pl
- gooses → goose + V + 3P + Sg
- dog → {dog + N + Sg, dog + V}
- leaves → {leaf + N + Pl, leave + V + 3P + Sg}

N: Noun; Pl: Plural; V: Verb; 3P: 3rd person; Sg: singular

Finite State Transducers

- Q: a finite set of states
- $q_0 \in Q$: a special start state
- $F \subseteq Q$: a set of final states
- Σ and Δ: two finite alphabets
- Transitions:

$s \in \Sigma^*, t \in \Delta^*$

- Encodes a set of strings that can be recognized by following paths from q_0 to some state in F

Tokenization

- Some Asian languages have no word boundary, e.g., Chinese
  - 语言学是一门关于人类语言的科学研究
- German too: Noun-noun compounds
  - *Gesundheitsversicherungsgesellschaften*
  - *Gesundheits-versicherungs-gesellschaften (health insurance companies)*
- Spanish clitics: *Dar-me-lo (To give me it)*
- Even English has issues, to a smaller degree: *Gregg and Bob's house*

Tokenization (Example)

Input raw text

```
Dr. Smith said tokenization of English is “harder than you’ve thought.”
When in New York, he paid $12.00 a day for lunch and wondered what it would
be like to work for AT&T or Google, Inc.
```

Output from Stanford Parser with Part-of-Speech tags: http://nlp.stanford.edu:8080/parser/index.jsp

```
Dr./NNP Smith/NNP said/VBD tokenization/NN of/IN English/NNP
is/VBZ ``/`` harder/JJR than/IN you/PRP 've/VBP thought/VBN ./.
''/''
When/WRB in/IN New/NNP York/NNP ,/, he/PRP paid/VBD $/$ 12.00/CD
a/DT day/NN for/IN lunch/NN and/CC wondered/VBD what/WP it/PRP
would/MD be/VB like/JJ to/TO work/VB for/IN AT&T/NNP or/CC
Google/NNP ,/, Inc./NNP ./.
```

Tokenization approaches

- **Traditional**: Segmenting words that make sense with grammars/meanings
  - For languages with word spaces: spaces, punctuation, plus rules
  - For Chinese etc: large dictionaries, punctuation, plus rules
- **Subword-based methods**: Segmenting words to max processing efficient/better
  - Split words into subword segments *without pre-tokenization or rules.*

Subword Tokenization

- Neural systems typically use a **relatively small fixed** vocabulary
- Real world contains many words
  - New words all the time
  - For morphologically rich languages, even more so
  - But most words are rare (Zipf's Law)
- Note that rare words do not have good corpus statistics
- So, tokenize words into more frequent subword segments

Unsupervised Subword Algorithms

- Use the data to tell us how to tokenize
- Three common algorithms:
  - **Byte-Pair Encoding (BPE)** [Sennrich et al., 2016]
  - **WordPiece** [Schuster and Nakajima, 2012]
  - **Unigram language modeling tokenization** (Unigram) [Kudo, 2018]
- Learnable tokenizer:
  - Training: takes a raw training corpus and induces a vocabulary
  - Segmentation: tokenizes a raw test sentence according to the vocabulary

BPE: https://github.com/rsennrich/subword-nmt
SentencePiece: https://github.com/google/sentencepiece

Byte-Pair Encoding

- Add a special end-of-word symbol "▁" (U+2581) or </w> at the end of each word in training corpus
- Convert words into a set of characters, create an initial vocabulary
- Iteratively merge the most frequent pair of adjacent tokens for k times

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

```
V ← all unique characters in C                         # initial set of tokens is characters
for i = 1 to k do                                      # merge tokens til k times
    t_L, t_R ← Most frequent pair of adjacent tokens in C
    t_NEW ← t_L + t_R                                  # make new token by concatenating
    V ← V + t_NEW                                      # update the vocabulary
    Replace each occurrence of t_L, t_R in C with t_NEW   # and update the corpus
return V
```

Byte-Pair Encoding (Example)

Example — training corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

low_ low_ low_ low_ low_ lowest_ lowest_ newer_ newer_ newer_ newer_ newer_ newer_ wider_ wider_ wider_ new_ new_

corpus	
5	l o w _
2	l o w e s t _
6	n e w e r _
3	w i d e r _
2	n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w

Byte-Pair Encoding (Example)

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w
2	l o w e s t _	
6	n e w e r _	
3	w i d e r _	
2	n e w _	

Merge e r to er

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2	l o w e s t _	
6	n e w er _	
3	w i d er _	
2	n e w _	

Byte-Pair Encoding (Example)

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2	l o w e s t _	
6	n e w er _	
3	w i d er _	
2	n e w _	

Merge er _ to er_

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2	l o w e s t _	
6	n e w er_	
3	w i d er_	
2	n e w _	

Byte-Pair Encoding (Example)

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2	l o w e s t _	
6	n e w er_	
3	w i d er_	
2	n e w _	

Merge n e to ne

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2	l o w e s t _	
6	ne w er_	
3	w i d er_	
2	ne w _	

Byte-Pair Encoding (Example)

- The next merges are:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

+: Usually include frequent words,
and frequent subwords which are often morphemes, e.g., *-est* or *-er*

Syntax
& Sequence Labeling

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*)
  - Nouns
  - Verbs
  - Adjectives
  - Adverbs

- **Closed classes** of words do not allow new members and usually involve grammatical rather than lexical words.
  - Prepositions
  - Determiners
  - Pronouns
  - Conjunctions
  - Auxiliary verbs

PART OF SPEECH	DT	VBZ	DT	JJ	NN
WORDS	This	is	a	simple	sentence

Part of speech tagsets

- 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	

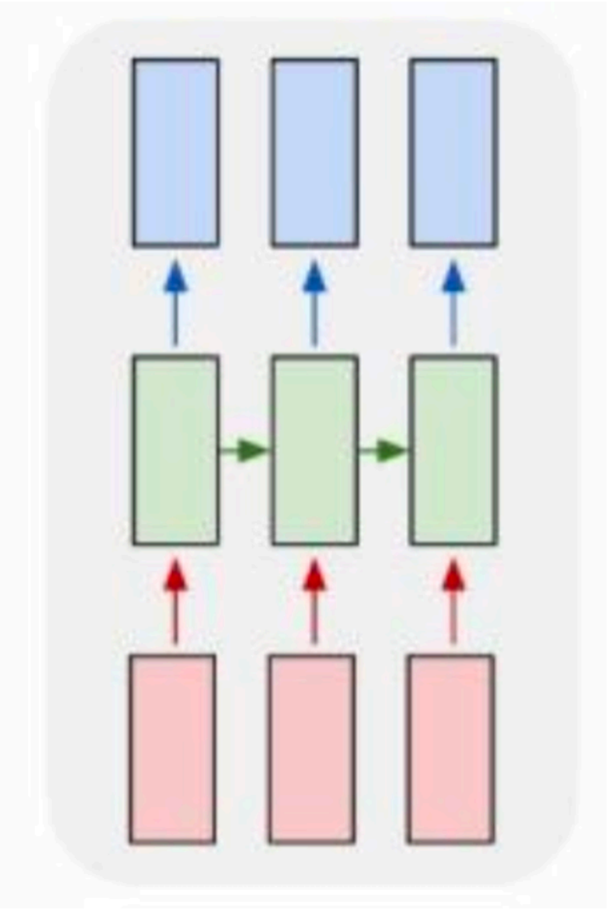
Why POS tagging?

- 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

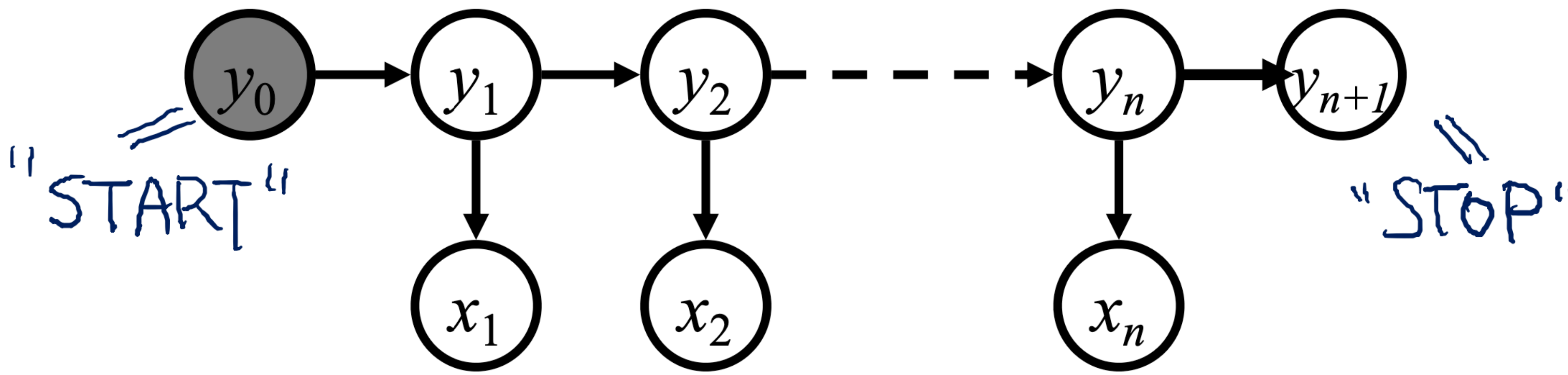
- Hidden Markov Models
- Conditional Random Fields
- Neural network-based methods

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



Classic Solution: HMMs

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



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

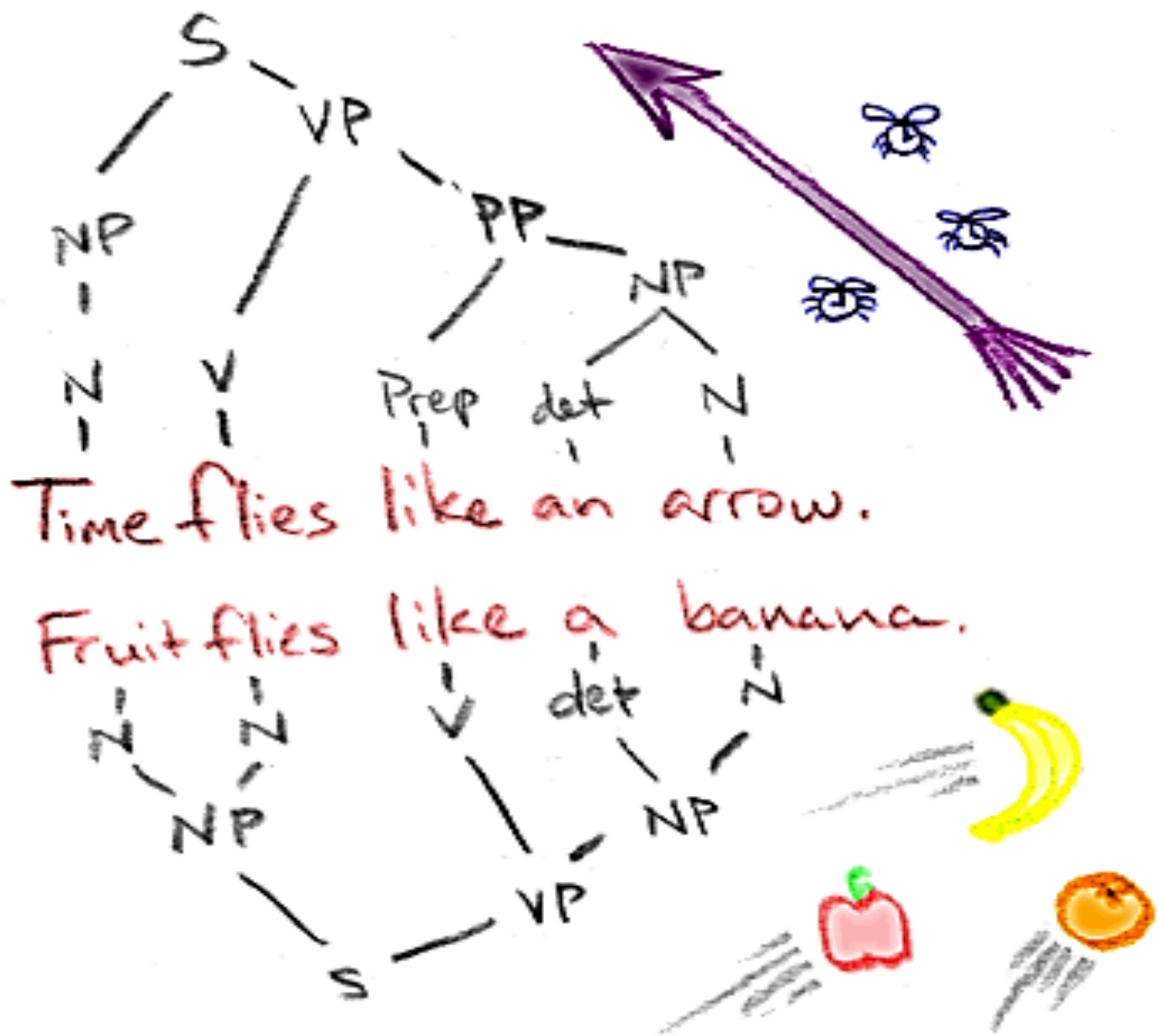
where $y_0 = \text{START}$ and we call $q(y'|y)$ the transition distribution and $e(x|y)$ the emission (or observation) distribution.

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 $e(x|y)$

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

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

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

$$\text{where } y_{n+1} = \text{STOP}$$

  - 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

Named Entity Recognition (Bickel et. al, 1999)

- Convert low-frequency words to word classes

Word class	Example	Intuition
twoDigitNum	90	Two digit year
fourDigitNum	1990	Four digit year
containsDigitAndAlpha	A8956-67	Product code
containsDigitAndDash	09-96	Date
containsDigitAndSlash	11/9/89	Date
containsDigitAndComma	23,000.00	Monetary amount
containsDigitAndPeriod	1.00	Monetary amount,percentage
othernum	456789	Other number
allCaps	BBN	Organization
capPeriod	M.	Person name initial
firstWord	first word of sentence	no useful capitalization information
initCap	Sally	Capitalized word
lowercase	can	Uncapitalized word
other	,	Punctuation marks, all other words

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	.

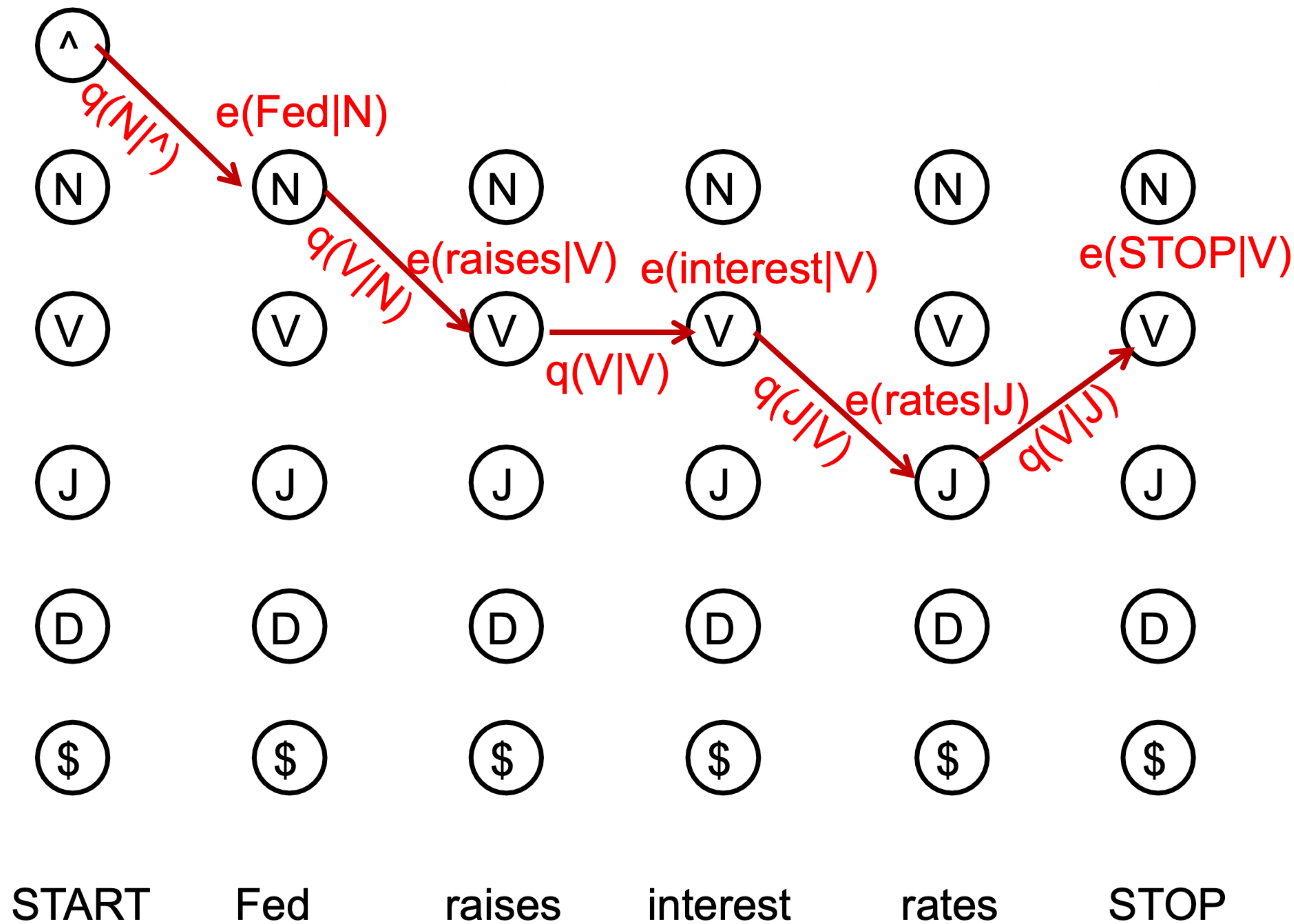
- In principle, we can list all possible tag sequences, score each one, and pick **the best one (a.k.a. the Viterbi state sequence)**

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

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

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

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

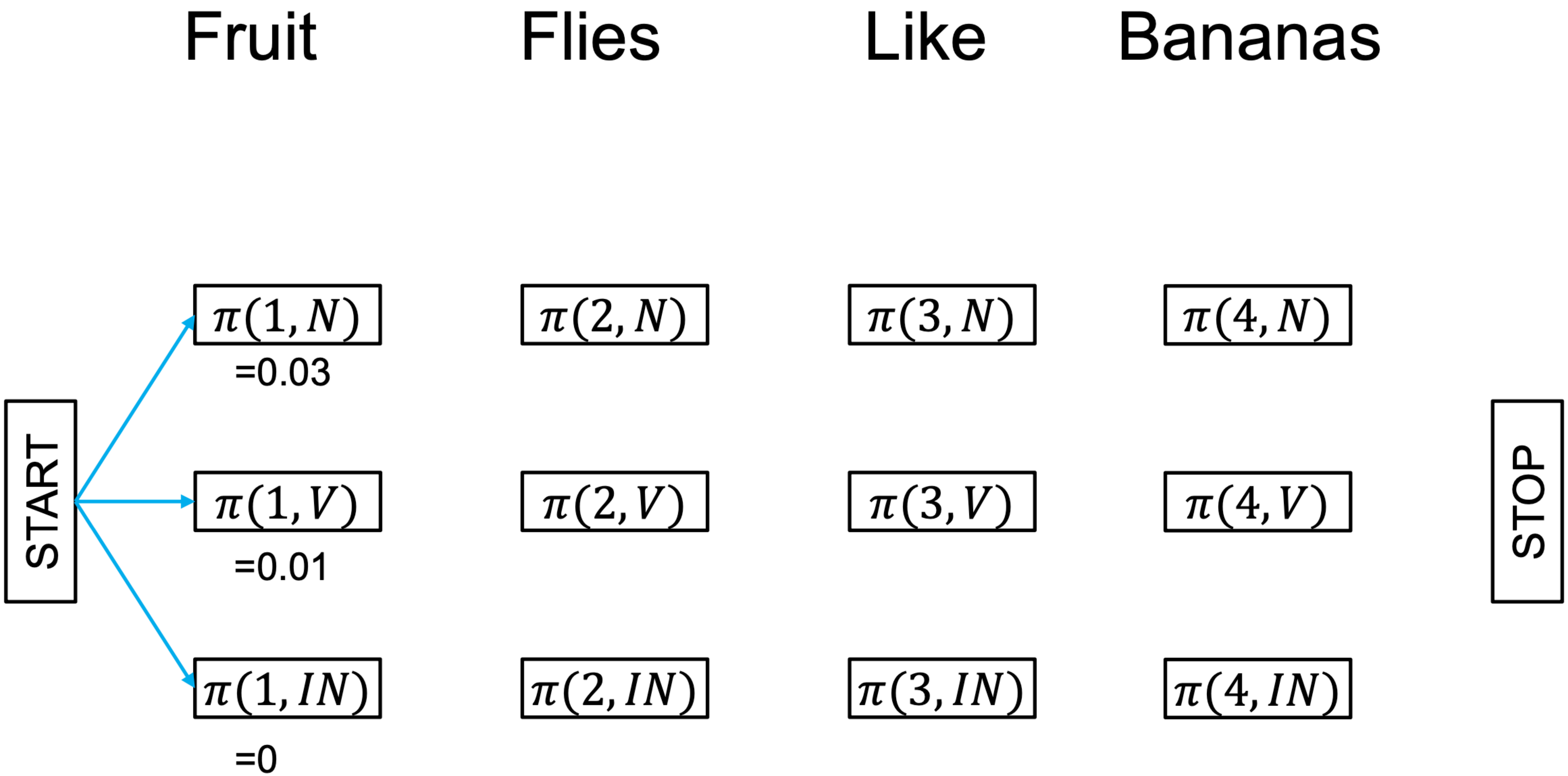
- We now have an efficient algorithm. Start with *i=0* and work your way to the end of the sentence!

Viterbi (Example)

	Fruit	Flies	Like	Bananas	
	$\pi(1,N)$	$\pi(2,N)$	$\pi(3,N)$	$\pi(4,N)$	
START	$\pi(1,V)$	$\pi(2,V)$	$\pi(3,V)$	$\pi(4,V)$	STOP
	$\pi(1,IN)$	$\pi(2,IN)$	$\pi(3,IN)$	$\pi(4,IN)$	

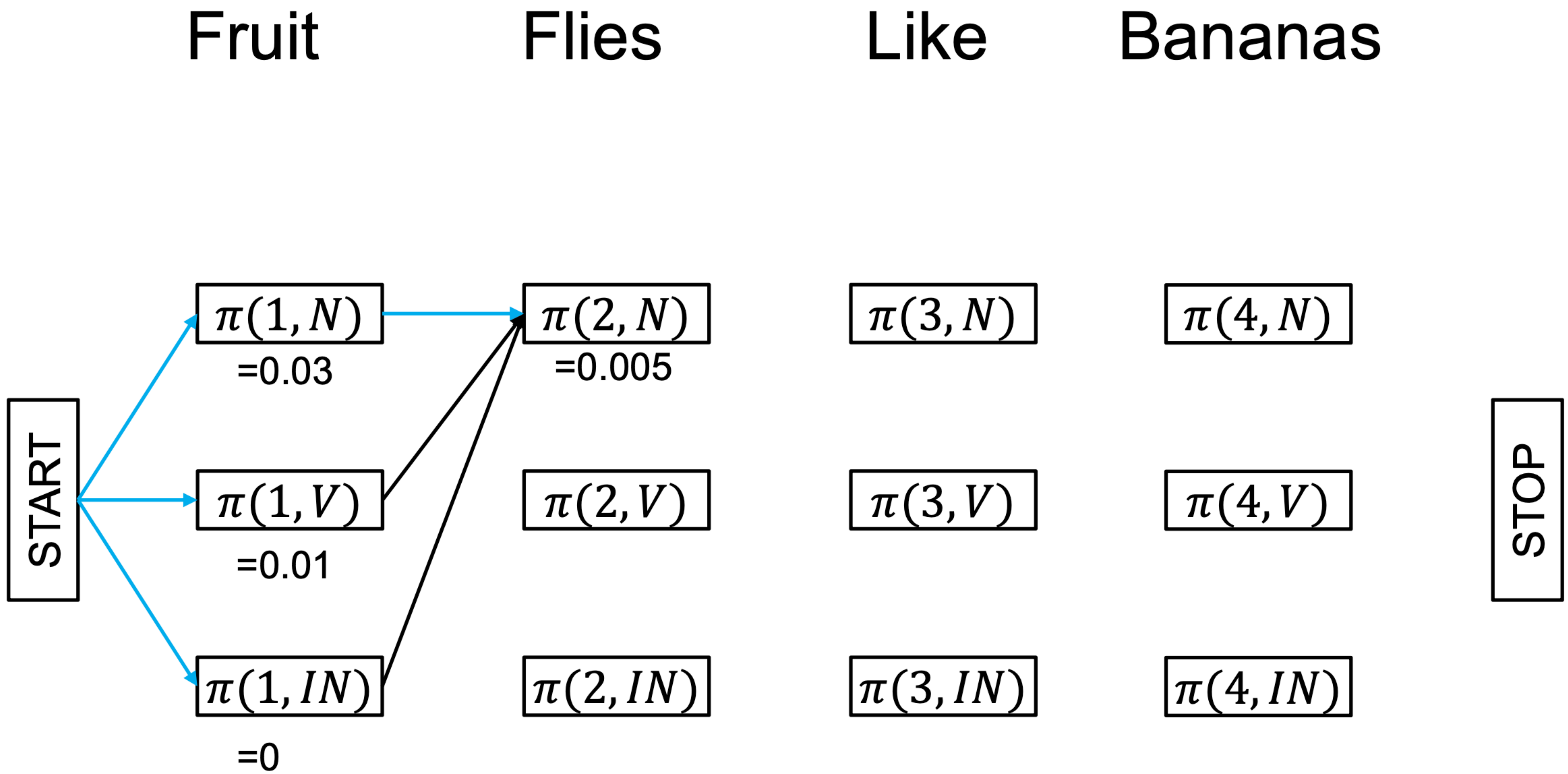
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

Viterbi (Example)



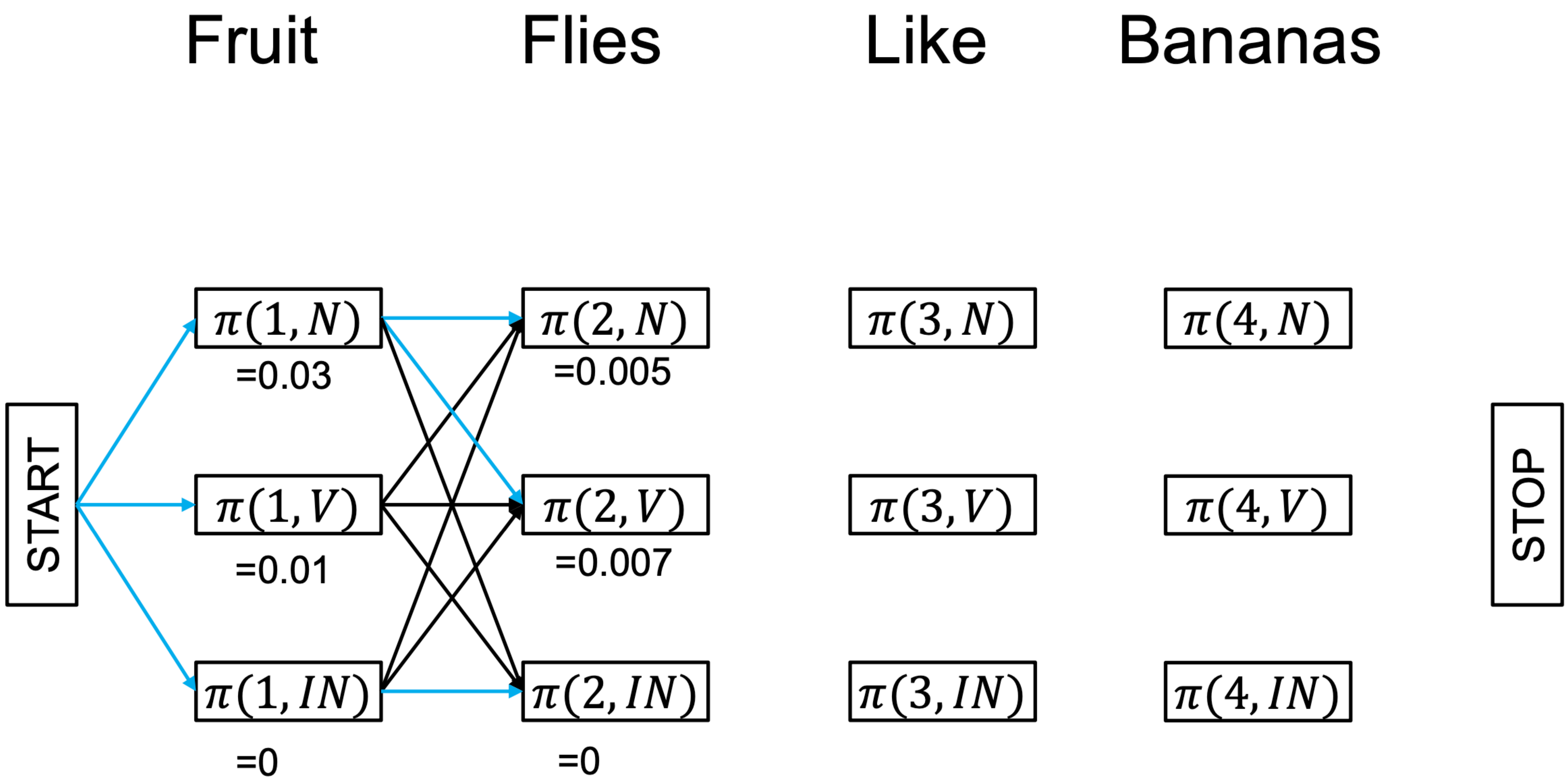
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

Viterbi (Example)



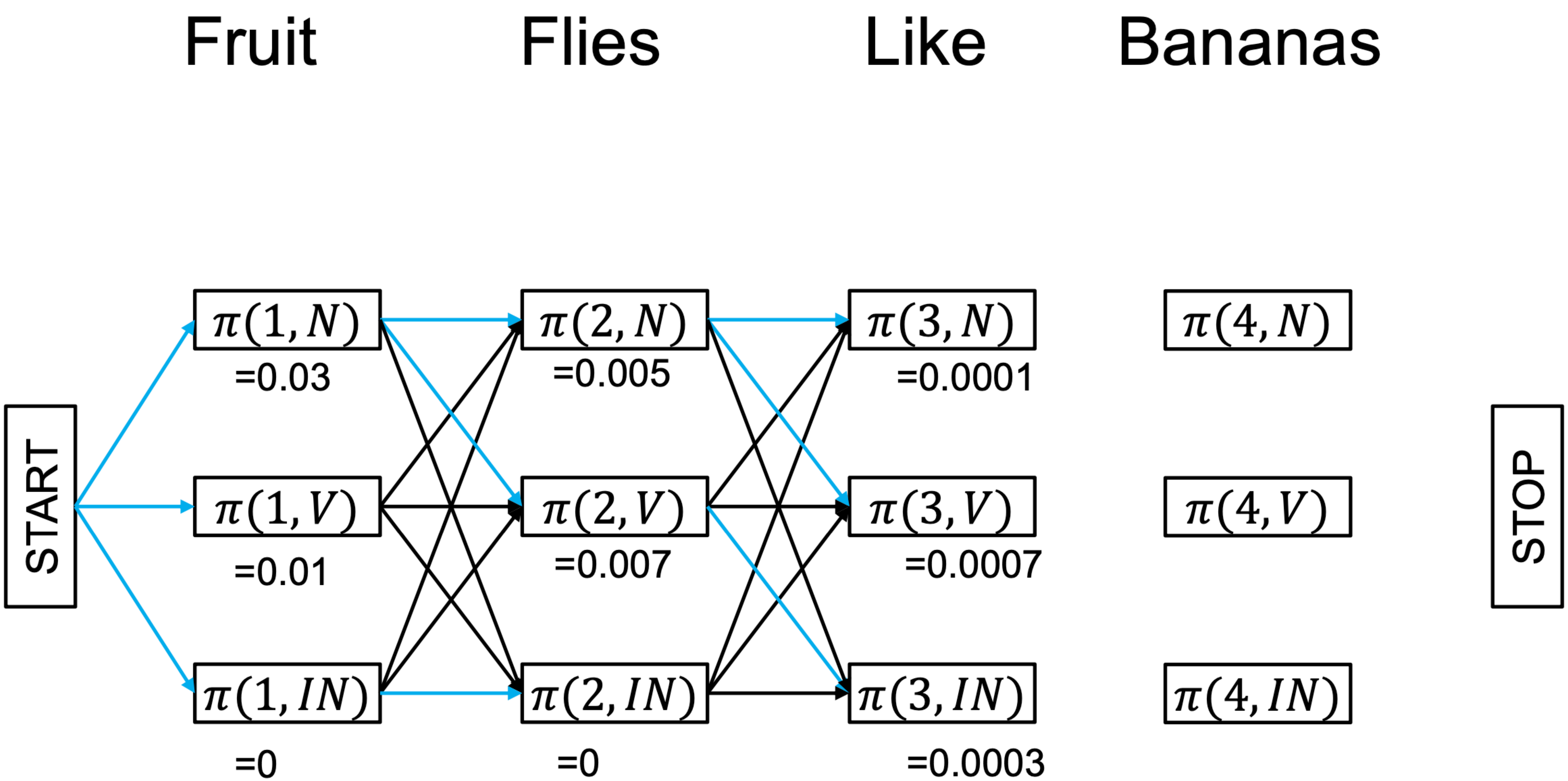
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

Viterbi (Example)



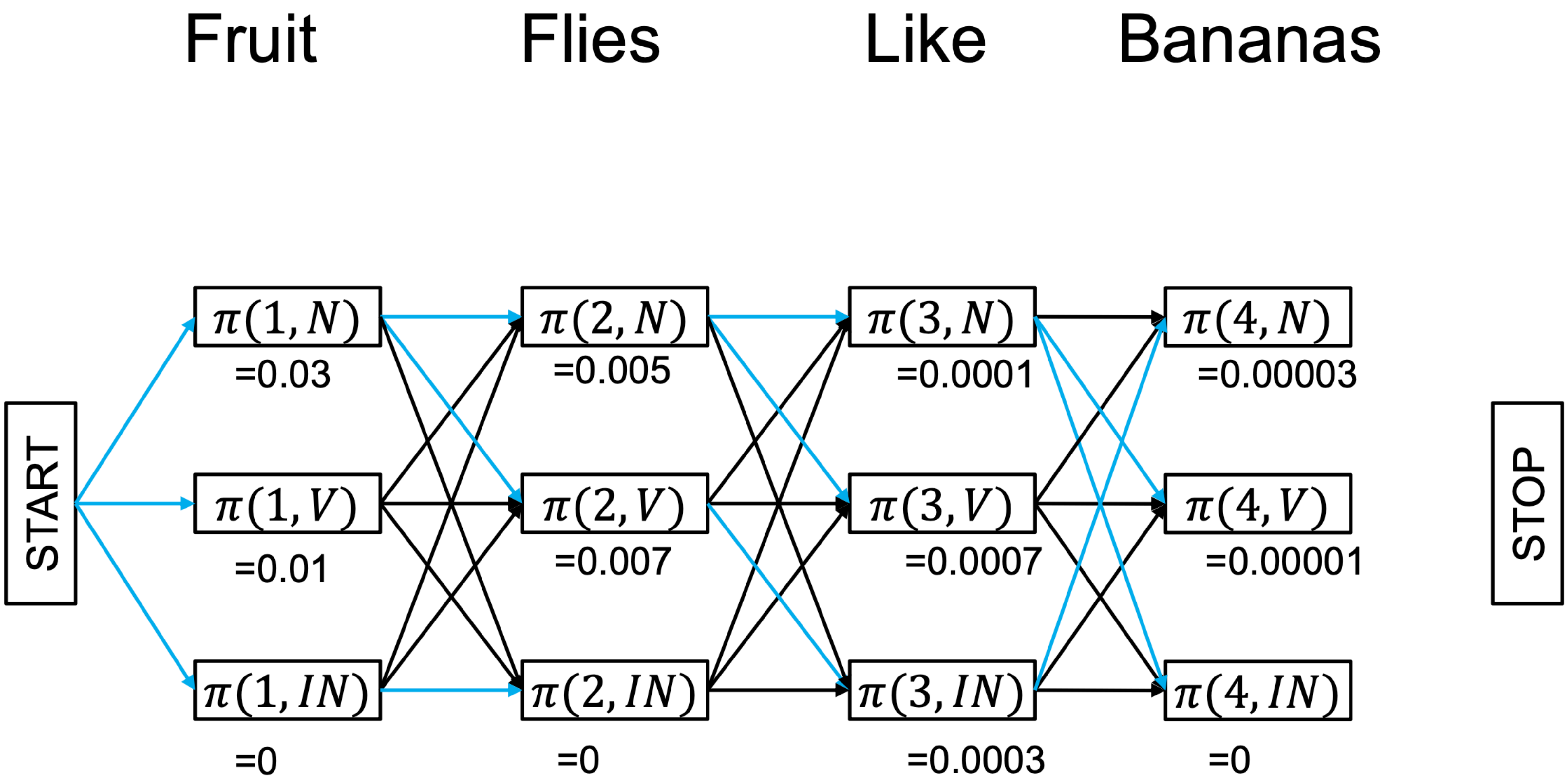
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

Viterbi (Example)



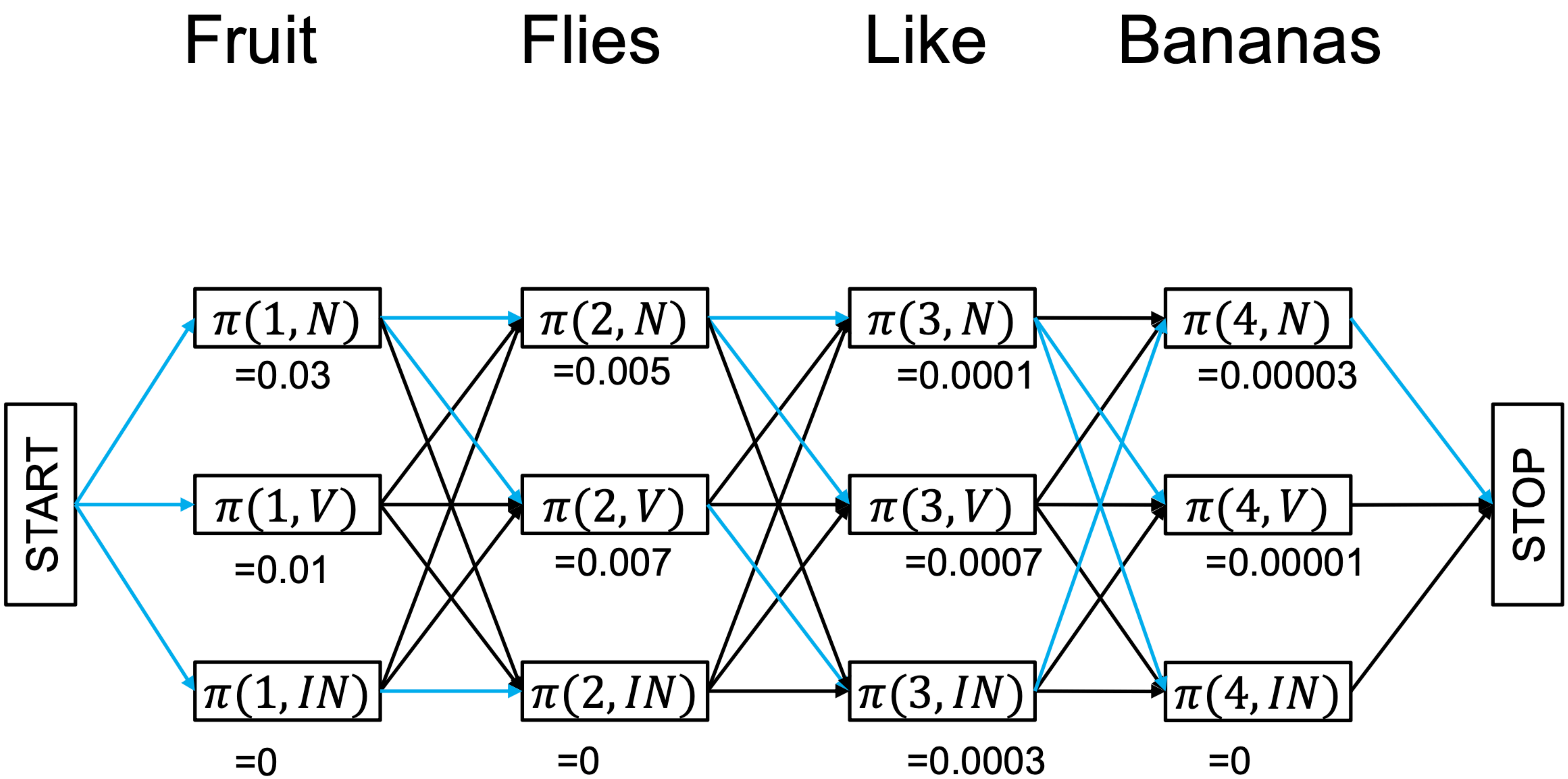
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \pi(i-1, y_{i-1})$$

Viterbi (Example)



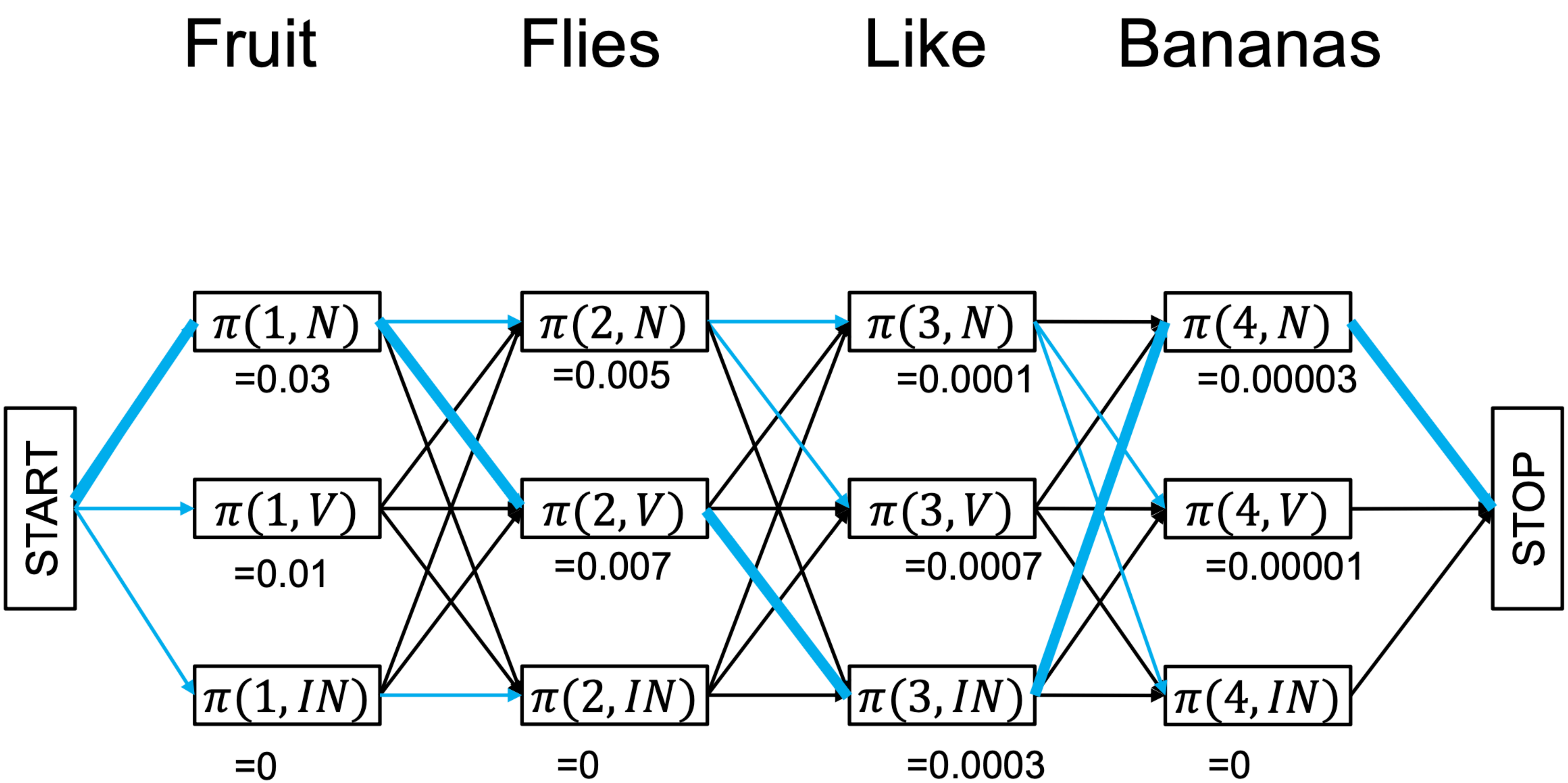
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \pi(i-1, y_{i-1})$$

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)



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

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

Viterbi Algorithm

- Dynamic programming (for all i)

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

  - 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 $\mathcal{K}$

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

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?