Announcement

- Midterm
  - Time: Mar. 21, in class
  - Location: this classroom
  - Format
    - Closed-book. You can bring an A4-size cheat sheet and nothing else.
  - Grade
    - 35% of the total grade

- Midterm review lecture on Mar. 16

Announcement

- Homework 3
  - Available in Blackboard -> Homework
  - Due: Mar. 19, 11:59pm (5 days)

Sequence Labeling

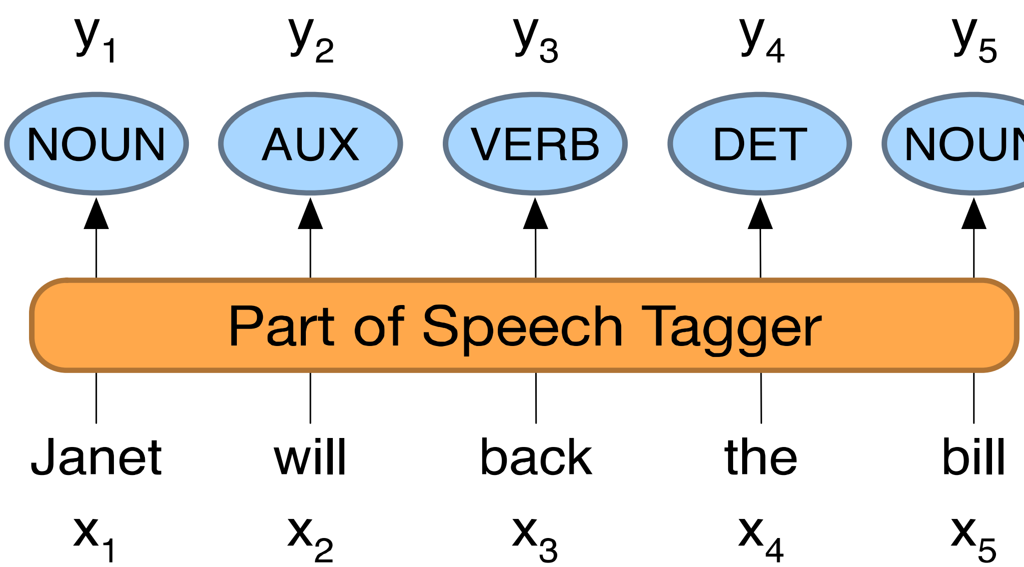
SLP3 Ch 8, 9.4; INLP Ch 7, 8

Sequence Labeling

- Known
  - A set of labels $Y = \{y^1, y^2, \cdots, y^n\}$
- Input:
  - Sentences $x = \{x_1, x_2, \ldots, x_m\}$
- Output:
  - For each word x_i, predict a label $y_i \in Y$

Part-of-Speech (POS) Tagging

- Map from sequence $x_1, x_2, \ldots, x_m$ of words to $y_1, y_2, \ldots, y_m$ of POS tags



Two classes of words: Open vs. Closed

- Closed class words
  - Relatively fixed for each language
  - Usually **function** words: short, frequent words with grammatical function
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
- Open class words
  - Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
    - Plus interjections: **oh, ouch, uh-huh, yes, hello**

Two classes of words: Open vs. Closed

Open class ("content") words

- Nouns
  - Proper: *Janet*, *Italy*
  - Common: *cat, cats*, *mango*
- Verbs
  - Main: *eat*, *went*
  - Auxiliary: *can*, *had*
- Adjectives *old green tasty*
- Adverbs *slowly yesterday*
- Interjections *Ow hello*
- Numbers: *122,312*, *one*
- *… more*

Closed class ("function")

- Determiners *the some*
- Conjunctions *and or*
- Pronouns *they its*
- Prepositions *to with*
- Particles *off up*
- *… more*

Why Part-of-Speech Tagging

- Can be useful for other NLP tasks
  - Parsing
    - POS tagging can improve syntactic parsing
  - MT
    - reordering of adjectives and nouns (say from Spanish to English)
  - Sentiment or affective tasks
    - may want to distinguish adjectives or other POS
  - Text-to-speech
    - how do we pronounce "lead" or "object"?

Other sequence labeling tasks

- Chinese word segmentation
  - Input

    瓦 里 西 斯 的 船 只 中 …
  - Output

    B I I E S B E S …

    (瓦 里 西 斯) (的) (船 只) (中) …

B = beginning of a word

I = inside of a word

E = end of a word

S = single character word

Other sequence labeling tasks

- Named entity recognition
  - Input

Michael	Jeffrey	Jordan	was	born	in	Brooklyn	…

  - Output

B-PER	I-PER	E-PER	O	O	O	S-LOC

Michael Jeffrey Jordan
Person

Brooklyn
Location

B = beginning of an entity
I = inside of an entity
E = end of an entity
S = single word entity
O = outside of any entity

-PER = person
-LOC = location
-ORG = organization
…

Other sequence labeling tasks

- Semantic role labeling
  - Input

The cat loves hats …

  - Output

B-ARG0 E-ARG0 S-PRED S-ARG1

<u>The cat</u> $\xleftarrow{\text{arg0}}$ loves $\xrightarrow{\text{arg1}}$ <u>hats</u>

B = beginning of an entity

I = inside of an entity

E = end of an entity

S = single word entity

O = outside of any entity

-PRED = predicate

-ARG0 = agent

-ARG1 = patient

…

The simplest method

- For each word, predict its most frequent label
  - 90% accuracy on POS tagging!
  - Disadvantages:
    1. It does not consider the contextual info
       - “book a flight” vs. “read a book”
       - 我骑车差点摔倒，好在我一把把把把住了
       - 校长说衣服上除了校徽别别别的
    2. It does not consider relations between adjacent labels
       - In BIOES: "B-I" and "B-E" are OK, but "B-O" and "B-S" are not

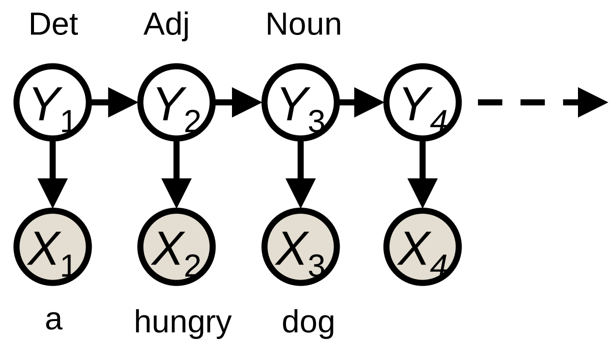
Method

- Hidden Markov model (HMM)
- Conditional random filed (CRF)
- Neural models

Hidden Markov Model

Hidden Markov Model (HMM)

- Variables
  - X: word
  - Y: label (hidden state)
- Parameters
  - Transition model $P(y_t|y_{t-1})$
    - Similar to a bigram model
  - Emission model $P(x_t|y_t)$
  - Initial distribution $P(y_1)$
    - Can be seen as transition from Y_0=START to Y_1
  - Modeling end of sequence
    - Can be seen as transition from Y_n to Y_{n+1}=STOP
- Joint prob: $P(x_1 \cdots x_n, y_1 \cdots y_{n+1}) = \prod_t P(y_t|y_{t-1}) P(x_t|y_t)$



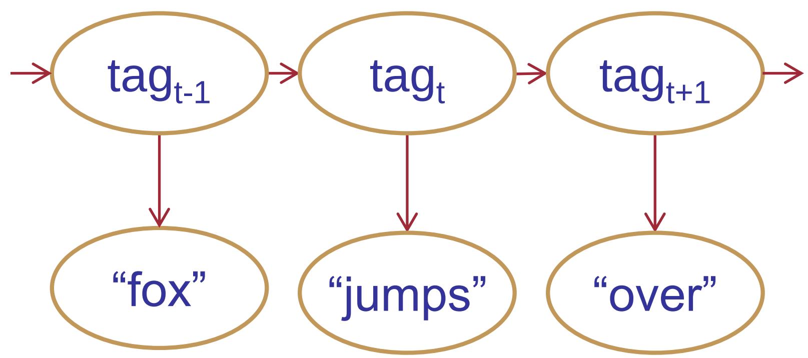
HMM Example

Transition

Y_{t-1}	$P(Y_t \mid Y_{t-1})$			
	N	V	P	…
START	0.5	0.1	0.1	…
N	0.4	0.3	0.1	…
V	0.5	0	0.3	…
P	0.3	0.1	0	…
…	…	…	…	…

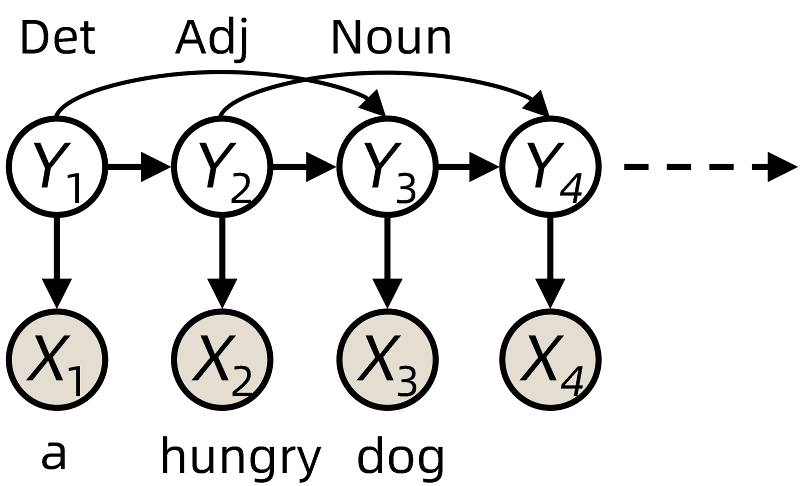
Emission

Y_t	$P(X_t \mid Y_t)$			
	"fox"	"dog"	"run"	…
N	0.02	0.03	0.01	…
V	0	0	0.05	…
P	0	0	0	…
…	…	…	…	…



High-order HMM

- Transition model $P(y_t | y_{t-1}, y_{t-2}, \cdots, y_{t-\mathrm{n}+1})$
  - Similar to an n-gram model



HMM Inference (Decoding)

- Given an input sequence, find the most likely label sequence under the model

$$y^* = \underset{y_1 \cdots y_n}{\operatorname{argmax}} P(y_1 \cdots y_{n+1} | x_1 \cdots x_n) = \underset{y_1 \cdots y_n}{\operatorname{argmax}} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

- Given an input, we can score any tag sequence
  - $P(x_1 \cdots x_n, y_1 \cdots y_{n+1}) = \prod_t P(y_t | y_{t-1}) P(x_t | y_t)$

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

q(NNP|START) e(Fed|NNP) q(VBZ|NNP) e(raises|VBZ) q(NN|VBZ)…..

HMM Inference (Decoding)

- Given an input sequence, find the most likely label sequence under the model

$$y^* = \underset{y_1 \cdots y_n}{\operatorname{argmax}} P\left(y_1 \cdots y_{n+1} | x_1 \cdots x_n \right) = \underset{y_1 \cdots y_n}{\operatorname{argmax}} P\left(x_1 \cdots x_n, y_1 \cdots y_{n+1} \right)$$

- Given an input, we can score any tag sequence
- In principle, we're done – list all possible tag sequences, score each one, pick the best one
  - Exponential time complexity!

NNP VBZ NN NNS CD NN	$\Rightarrow$	logP = -23
NNP NNS NN NNS CD NN	$\Rightarrow$	logP = -29
NNP VBZ VB NNS CD NN	$\Rightarrow$	logP = -27
… …		… …

Dynamic Programming (Viterbi Algorithm)

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

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

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

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

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

Dynamic Programming (Viterbi Algorithm)

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

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

- We now have an efficient DP algorithm
  - Start with $\pi(0, \mathrm{START}) = 1$
  - Work your way to the end of the sentence

$$P(y^*) = \max_{y_1 \cdots y_n} P\left(x_1 \cdots x_n, y_1 \cdots y_{n+1}\right)$$

$$= \max_{y_n} q(STOP|y_n) \max_{y_1 \cdots y_{n-1}} P\left(x_1 \cdots x_n, y_1 \cdots y_n\right)$$

$$= \max_{y_n} q(STOP|y_n)\, \pi(n, y_n) \;\coloneqq\; \pi(n+1, \mathrm{STOP})$$

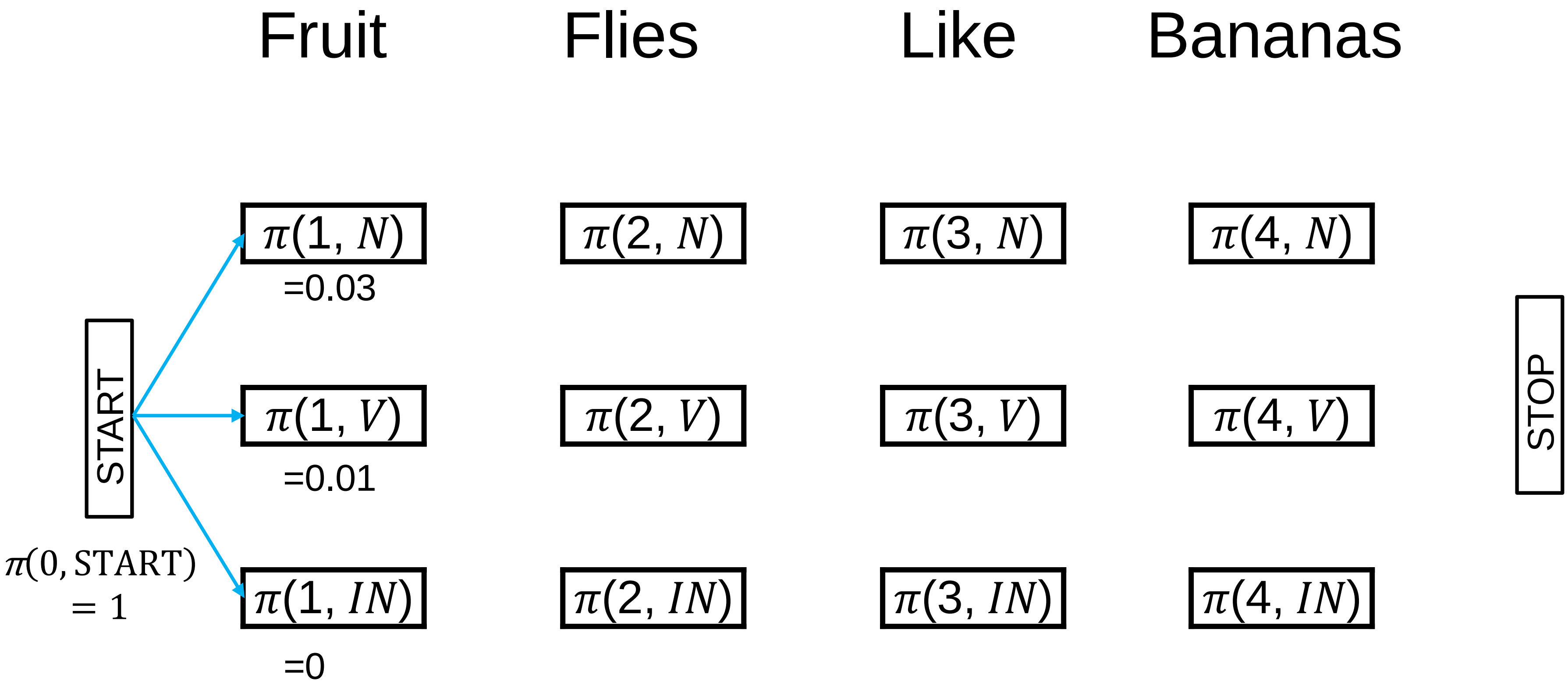
Example

State Trellis

	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
$\pi(0, \text{START}) = 1$	$\pi(1, IN)$	$\pi(2, IN)$	$\pi(3, IN)$	$\pi(4, IN)$	

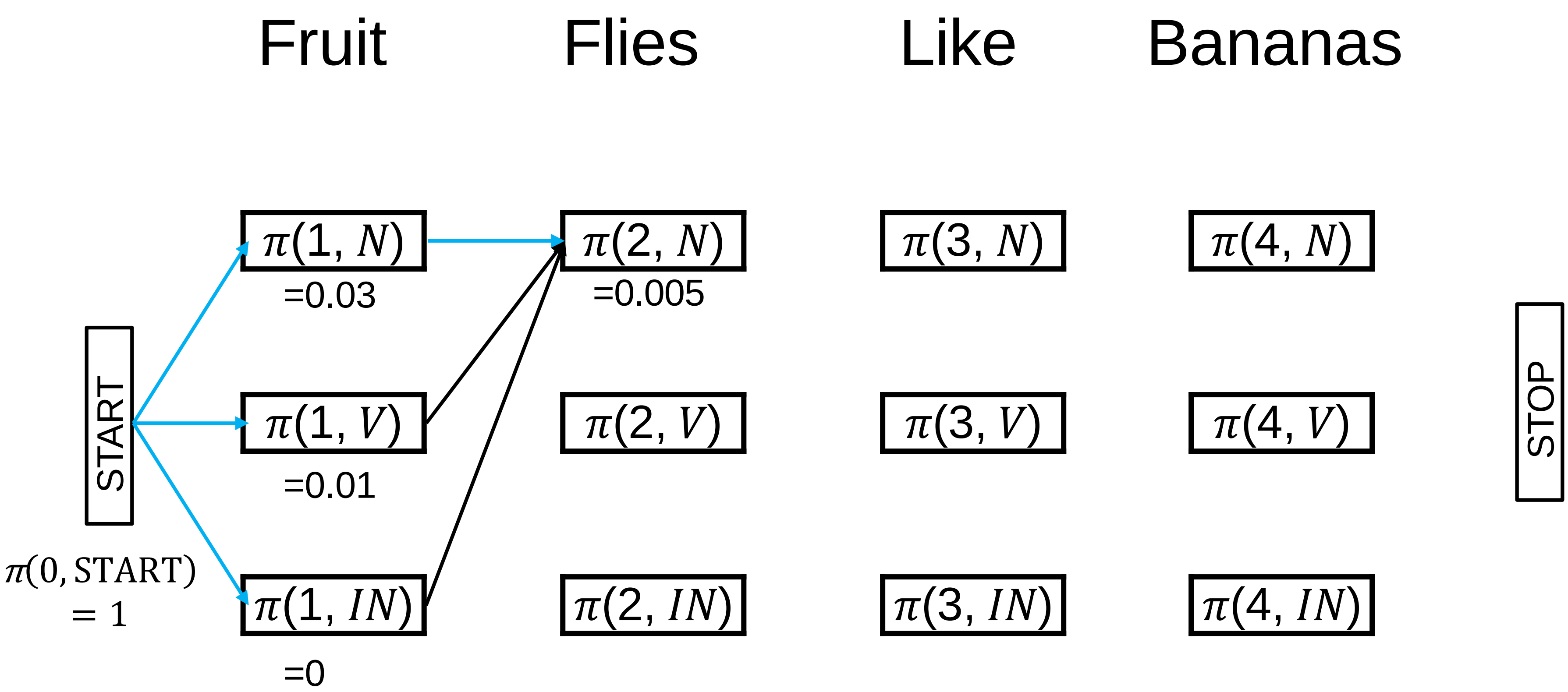
$$\pi(i, y_i) = e(x_i | y_i) \max_{y_{i-1}} q(y_i | y_{i-1})\, \pi(i-1, y_{i-1})$$

Example



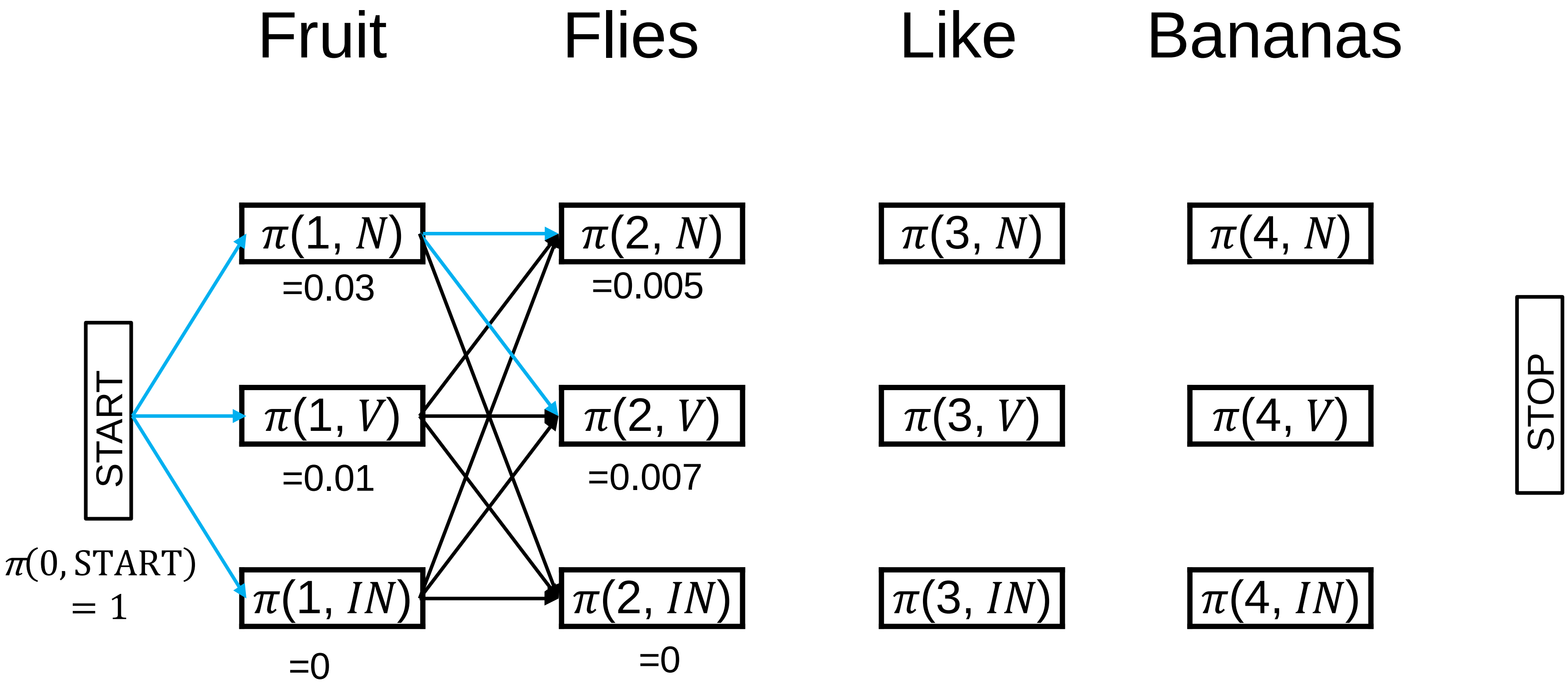
$$\pi(i, y_i) = e(x_i | y_i) \max_{y_{i-1}} q(y_i | y_{i-1}) \, \pi(i-1, y_{i-1})$$

Example



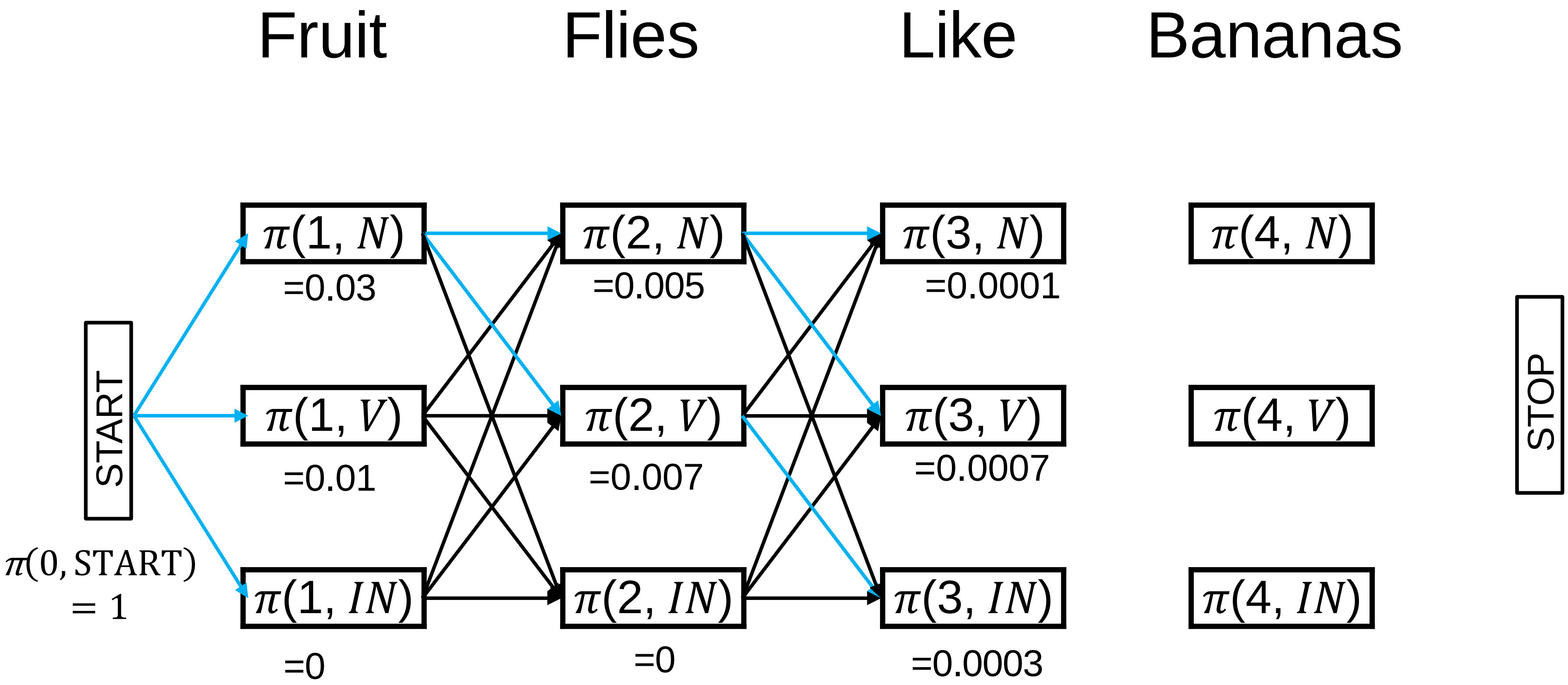
$$\pi(i, y_i) = e(x_i | y_i) \max_{y_{i-1}} q(y_i | y_{i-1})\, \pi(i-1, y_{i-1})$$

Example



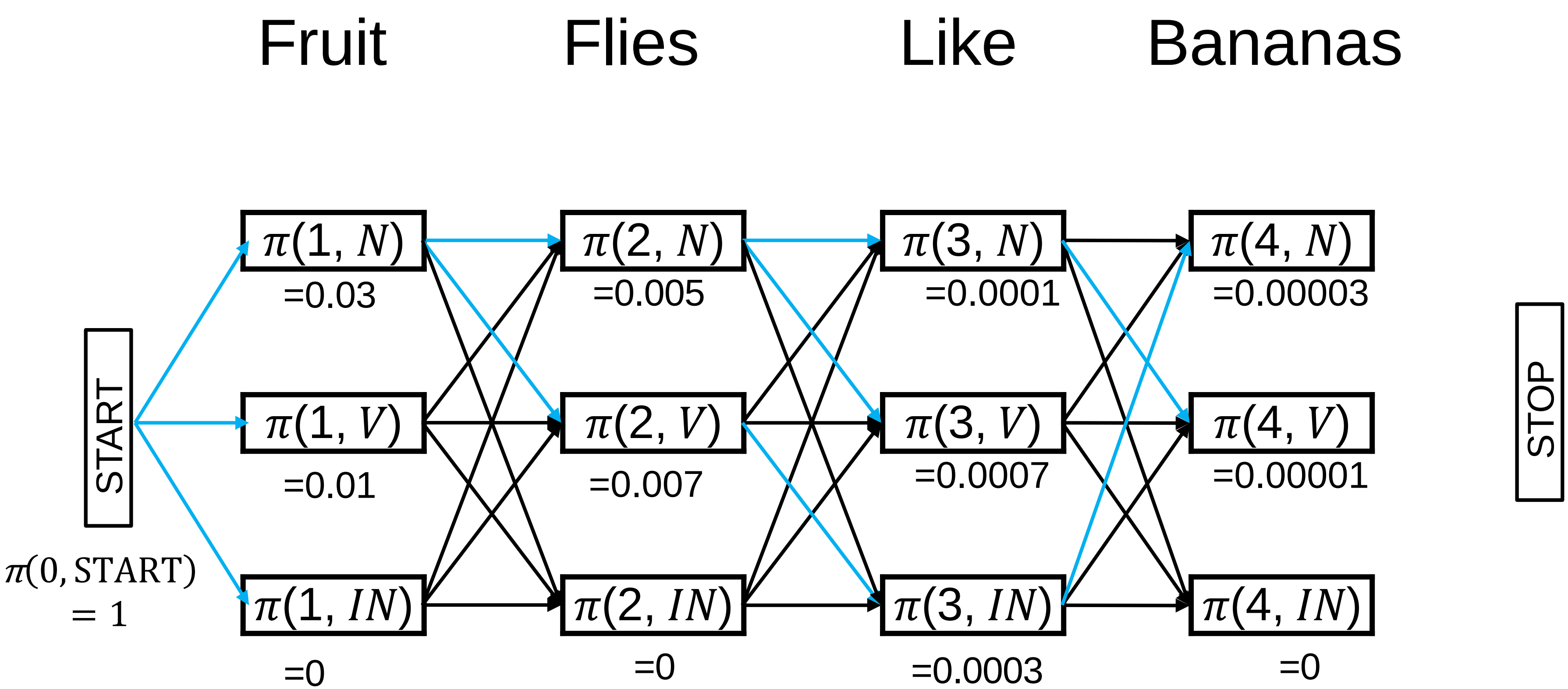
$$\pi(i, y_i) = e(x_i|y_i) \max_{y_{i-1}} q(y_i|y_{i-1})\, \pi(i-1, y_{i-1})$$

Example



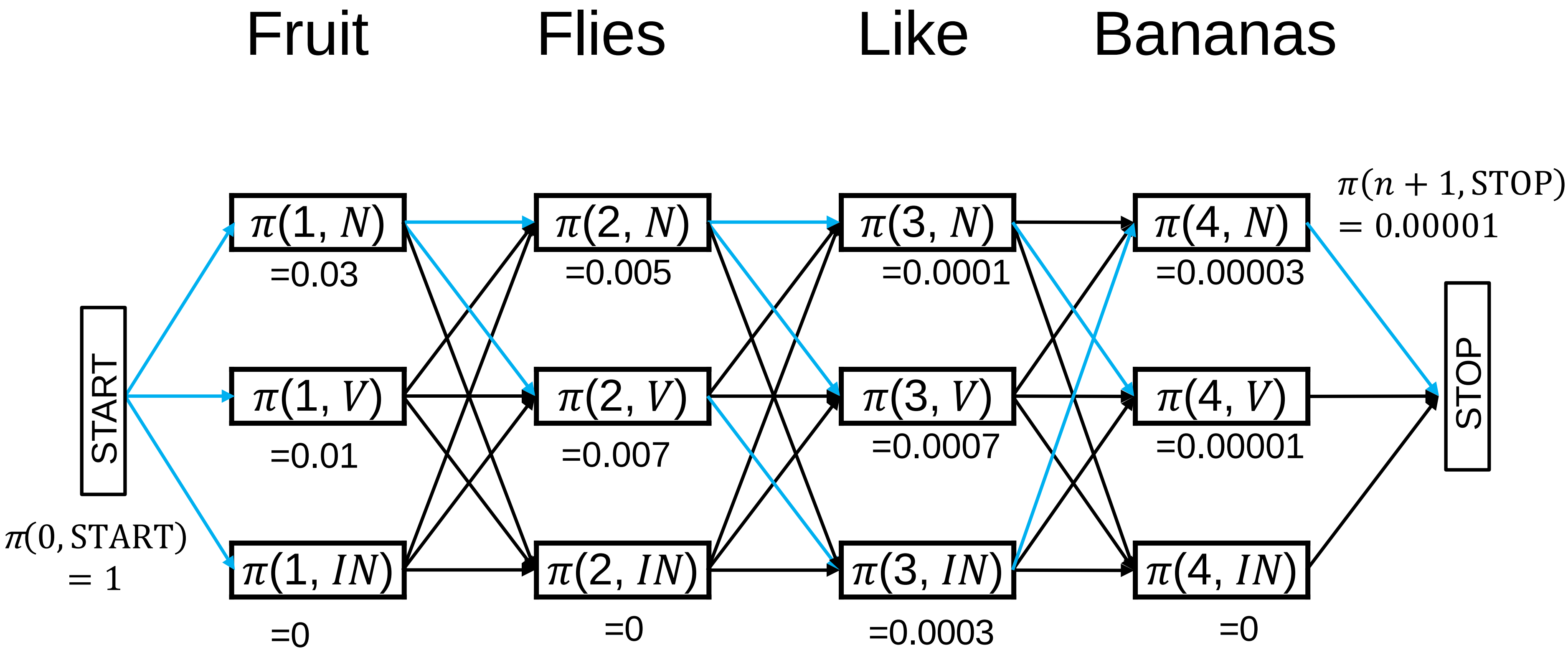
$$\pi(i, y_i) = e(x_i | y_i) \max_{y_{i-1}} q(y_i | y_{i-1}) \, \pi(i-1, y_{i-1})$$

Example



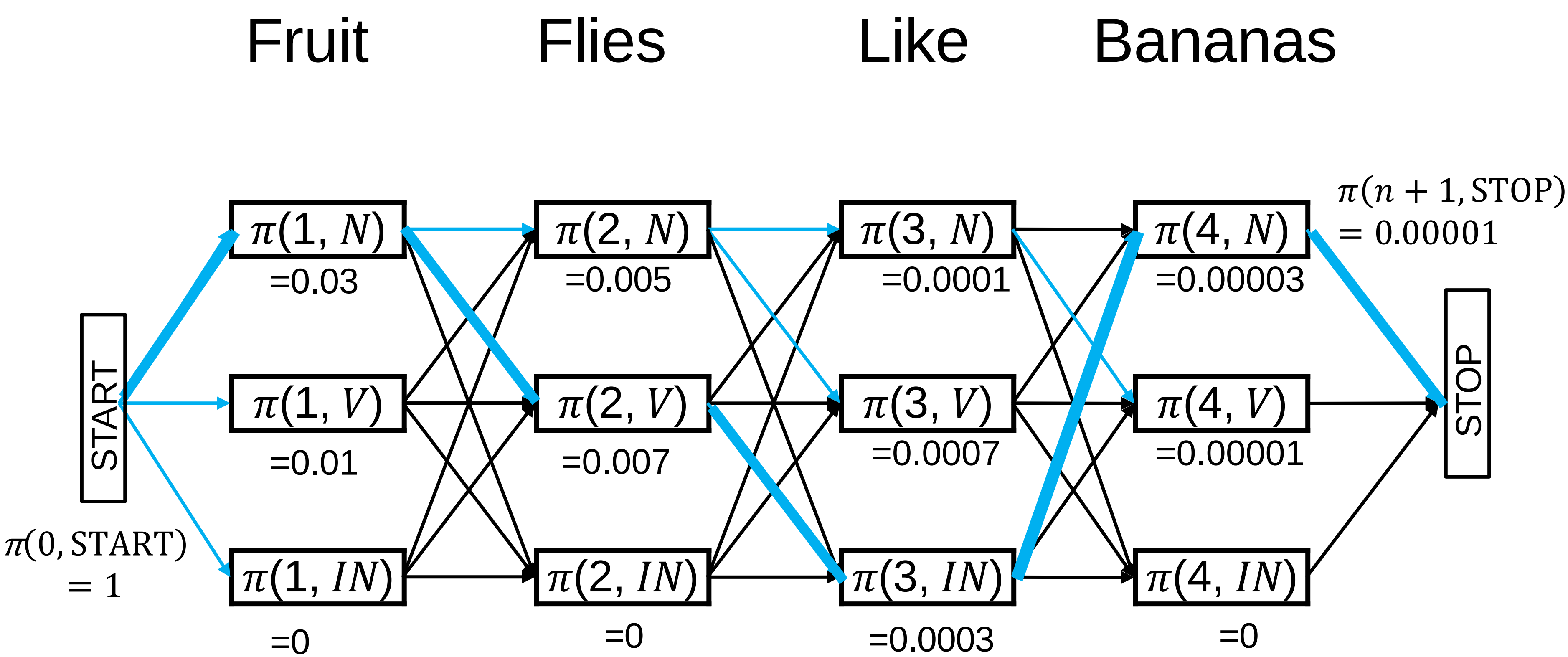
$$\pi(i, y_i) = e(x_i|y_i) \max_{y_{i-1}} q(y_i|y_{i-1})\, \pi(i-1, y_{i-1})$$

Example



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

Example



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

The Viterbi Algorithm: Runtime

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

- Sentence length *n*, tag number |Y|
- $O(n|Y|)$ entries in $\pi(i, y_i)$
- $O(|Y|)$ time to compute each $\pi(i, y_i)$
- Total runtime: $O(n|Y|^2)$

Marginal Inference

- Compute the marginal probability of the input sentence
  - $P(x_1 \cdots\ x_n) = \sum_{y_1 \dots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$
- Given an input, we can score any tag sequence
- In principle, we're done – list all possible tag sequences with y_i , score each one, take summation
  - Exponential time complexity!

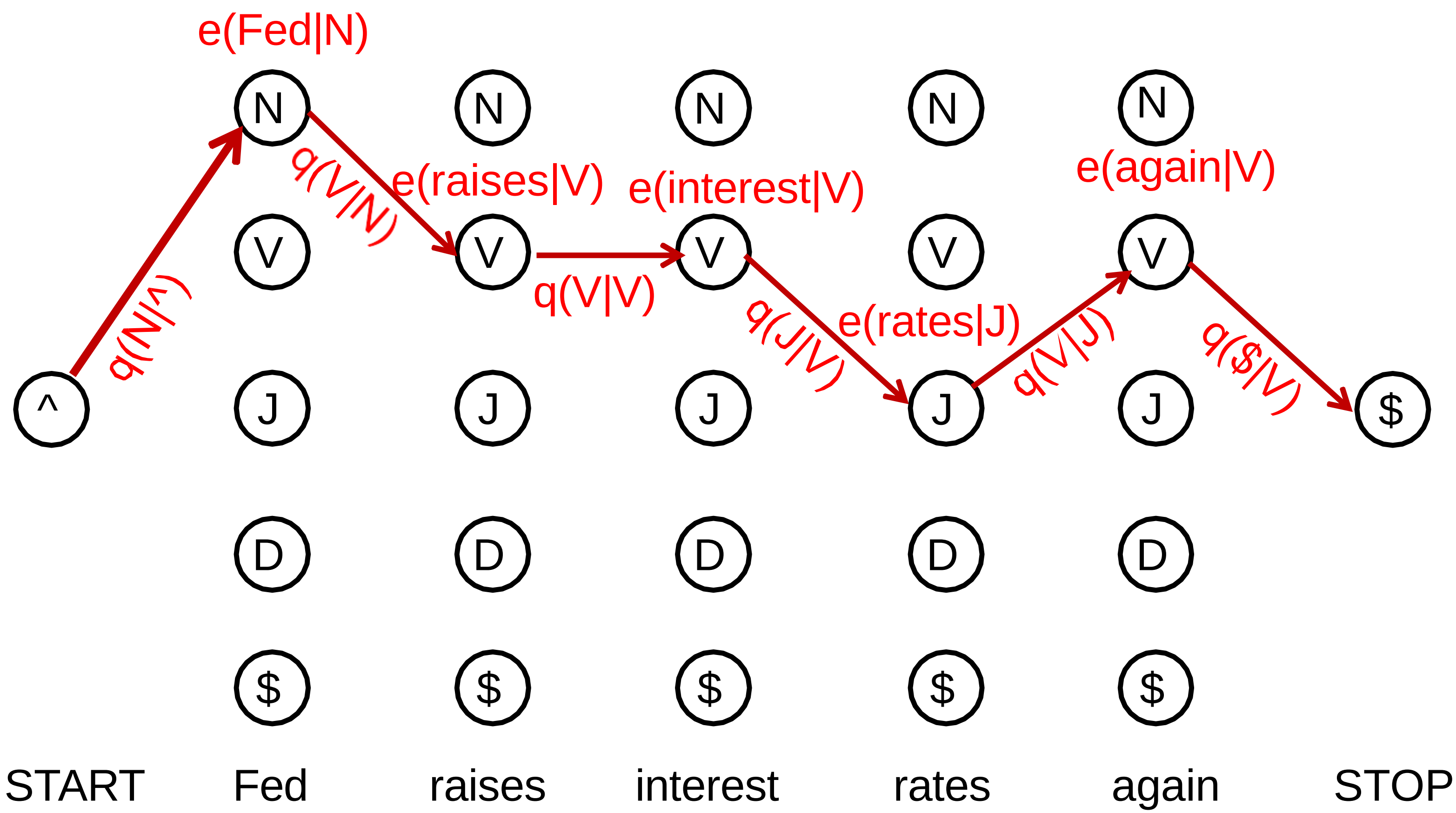
NNP VBZ NN NNS CD NN	logP = -23
NNP NNS NN NNS CD NN	logP = -29
NNP VBZ VB NNS CD NN	logP = -27
… …	… …

Marginal Inference

- Compute the marginal probability of the input sentence
  - $P(x_1 \cdots x_n) = \sum_{y_1 \ldots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$
- Compare it with decoding
  - $y^* = \underset{y_1 \cdots y_n}{\operatorname{argmax}} P\left(x_1 \cdots x_n, y_1 \cdots y_{n+1}\right)$
  - $P(y^*) = \max_{y_1 \cdots y_n} P\left(x_1 \cdots x_n, y_1 \cdots y_{n+1}\right)$

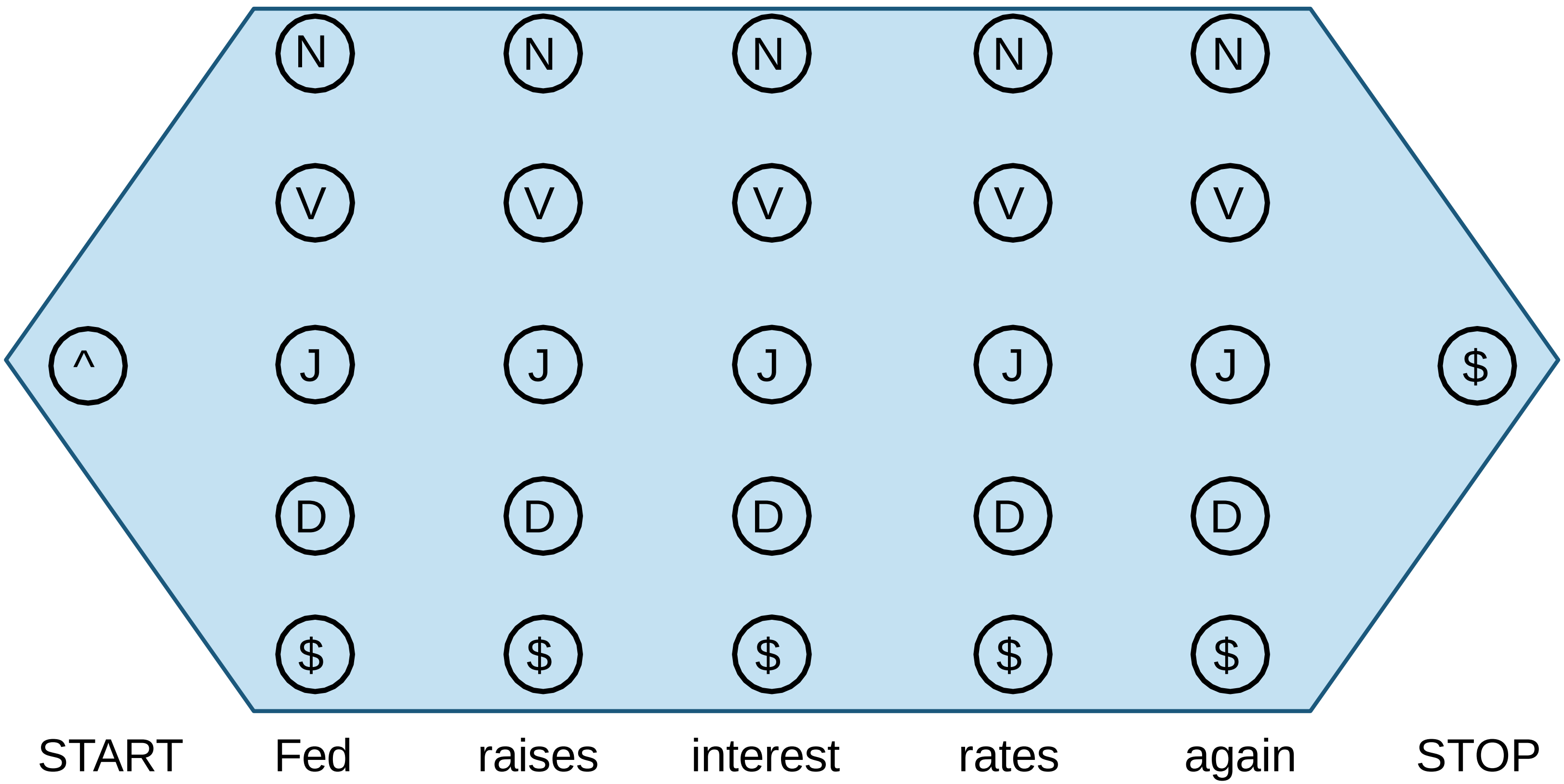
The State Trellis: Viterbi

$$\max_{y_1 \cdots y_n} P\left(x_1 \cdots x_n, y_1 \cdots y_{n+1}\right)$$



The State Trellis: Marginal

$$\sum_{y_1 \cdots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$



Dynamic Programming (Forward Algorithm)

$$\alpha(i, y_i) = P(x_1 \cdots x_i, y_i) = \sum_{y_1,\cdots,y_{i-1}} P(x_1 \cdots x_i, y_1 \cdots y_i)$$

$$= \sum_{y_1,\cdots,y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\ P(x_1 \cdots x_{i-1}, y_1 \cdots y_{i-1})$$

$$= \sum_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1}) \sum_{y_1,\cdots,y_{i-2}} P(x_1 \cdots x_{i-1}, y_1 \cdots y_{i-1})$$

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

Dynamic Programming (Forward Algorithm)

- Start with:

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

- For $i = 1, \cdots, n$:

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

- Finally:

$$P(x_1 \cdots x_n) = \sum_{y_1 \cdots y_n} P\,(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

$$= \sum_{y_n} q(STOP|y_n) \sum_{y_1 \cdots y_{n-1}} P\,(x_1 \cdots x_n, y_1 \cdots y_n)$$

$$= \sum_{y_n} q(STOP|y_n) \alpha(n, y_n) \; \coloneqq \alpha(n+1, STOP)$$

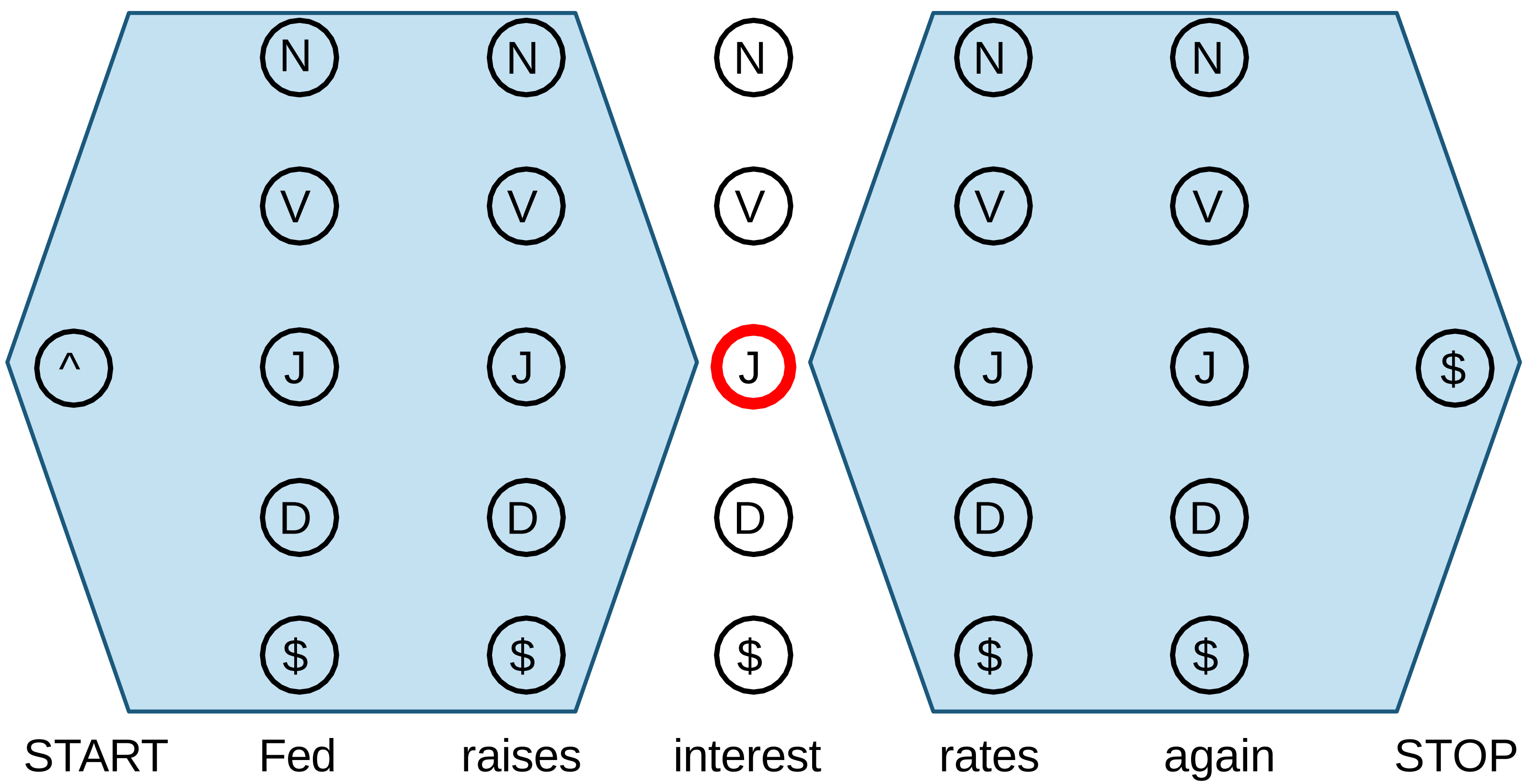
Marginal Inference

- Find the marginal probability of each tag for y_i
  - $P(x_1 \cdots x_n, y_i) = \sum_{y_1 \cdots y_{i-1}} \sum_{y_{i+1} \cdots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$
- Given an input, we can score any tag sequence
- In principle, we're done – list all possible tag sequences with y_i , score each one, take summation
  - Exponential time complexity!

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 Trellis: Marginal

$$\sum_{y_1 \dots y_{i-1}} \sum_{y_{i+1} \dots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

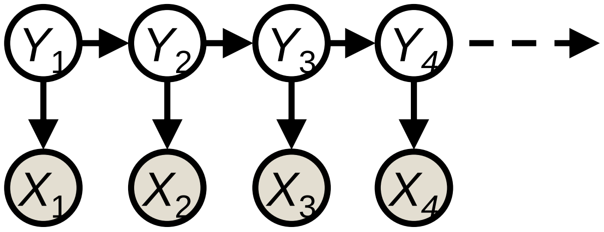


Dynamic Programming

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

$$P(x_1 \cdots x_n, y_i) = P(x_1 \cdots x_i, y_i)\, P(x_{i+1} \cdots x_n | y_i, x_1 \cdots x_i)$$

$$= \underbrace{P(x_1 \cdots x_i, y_i)}_{\alpha(i, y_i)}\, \underbrace{P(x_{i+1} \cdots x_n | y_i)}_{\beta(i, y_i)}$$



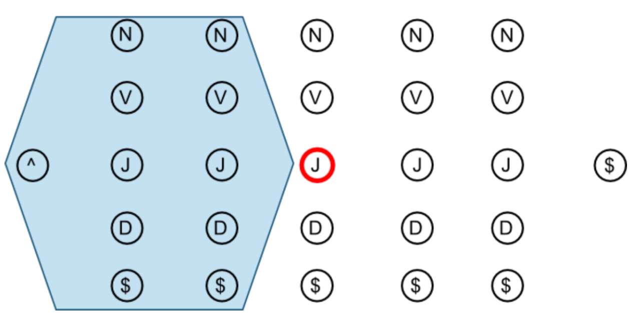
Forward

$$\alpha(i, y_i) = P(x_1 \cdots x_i, y_i) = \sum_{y_1, \cdots, y_{i-1}} P(x_1 \cdots x_i, y_1 \cdots y_i)$$

$$= \sum_{y_1, \cdots, y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \; P(x_1 \cdots x_{i-1}, y_1 \cdots y_{i-1})$$

$$= \sum_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \sum_{y_1, \cdots, y_{i-2}} P(x_1 \cdots x_{i-1}, y_1 \cdots y_{i-1})$$

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



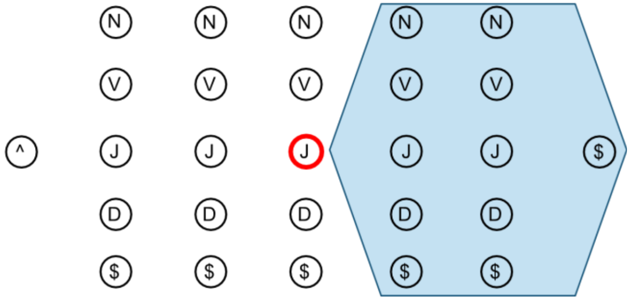
Backward

$$\beta(i, y_i) = P(x_{i+1} \cdots x_n | y_i) = \sum_{y_{i+1}, \cdots, y_n} P(x_{i+1} \cdots x_n, y_{i+1} \cdots y_{n+1} | y_i)$$

$$= \sum_{y_{i+1}, \cdots, y_n} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) P(x_{i+2} \cdots x_n, y_{i+2} \cdots y_{n+1} | y_{i+1})$$

$$= \sum_{y_{i+1}} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) \sum_{y_{i+2}, \cdots, y_n} P(x_{i+2} \cdots x_n, y_{i+2} \cdots y_{n+1} | y_{i+1})$$

$$= \sum_{y_{i+1}} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) \beta(i+1, y_{i+1})$$



Forward-Backward Algorithm

- Two passes: one forward, one backward
  - Forward

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

    - For $i = 1, \cdots, n$:

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

  - Backward

$$\beta(n, y_n) = \begin{cases} q(y_{n+1}|y_n)\ if\ y_{n+1} = STOP \\ 0 \qquad\qquad\qquad\quad otherwise \end{cases}$$

    - For $i = n-1, \cdots, 0$

$$\beta(i, y_i) = \sum_{y_{i+1}} e(x_{i+1}|y_{i+1})q(y_{i+1}|y_i)\beta(i+1, y_{i+1})$$

Forward-Backward: Runtime

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

$$\beta(i, y_i) = \sum_{y_{i+1}} e(x_{i+1}|y_{i+1})q(y_{i+1}|y_i)\beta(i+1, y_{i+1})$$

- Sentence length n, tag number |Y|
- $O(n|Y|)$ entries in $\alpha(i, y_i)$ and $\beta(i, y_i)$
- $O(|Y|)$ time to compute each entry
- Total runtime: $O(n|Y|^2)$
- Exactly the same as Viterbi

HMM Supervised Learning

- Learn HMM given annotated sequence $\{(x_1, y_1), \cdots, (x_n, y_n)\}$
  - Maximum likelihood estimate

$$P(x, y) = \prod_{i=1}^{n+1} e(x_i|y_i) \cdot q(y_i|y_{i-1}) = \prod_{i,j \in Y} q(j|i)^{c(i,j)} \prod_{j \in X} \prod_{i \in Y} e(j|i)^{c(i,j)}$$

e: emission; q: transition; c: co-occurrence count

  - Closed-form solution: count and normalize

$$e(k|i) = \frac{c(i,k)}{\sum_{k' \in X} c(i,k')} \qquad q(j|i) = \frac{c(i,j)}{\sum_{j' \in Y} c(i,j')}$$

  - Handle data sparseness
    - We can use all of the tricks we use for n-gram models

HMM Unsupervised Learning

- Learn HMM given unannotated sequence $\{x_1, \cdots, x_n\}$
- Application: part-of-speech induction
  - Induce the set of POS tags from text
- Maximize marginal likelihood

$$P(x_1 \cdots x_n) = \sum_{y_1 \cdots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

From “05 Text Clustering”

Expectation-Maximization (EM)

- Can be used to learn any model with hidden variables (missing data)
- Alternate:
  - Compute distributions over hidden variables based on current parameter values
  - Compute new parameter values to maximize expected log likelihood based on distributions over hidden variables
- Stop when no changes

- Can reach a local optimum but not necessarily a global optimum

EM for HMM (Baum-Welch Algorithm)

- Initialize transition and emission parameters
  - Random, uniform, or more informed initialization
- Iterate until convergence
  - E-Step:
    - Compute expected counts
    - General form:

$$c(S) = \mathrm{E}_{P(y_1 \cdots y_{n+1} | x_1 \cdots x_n)}[Count(S | x_1, \cdots, x_n, y_1 \cdots y_{n+1})]$$

    - Computing:

$$c(NN) = \sum_i P(y_i = NN \,| x_1, \cdots, x_n)$$

$$c(NN \rightarrow VB) = \sum_i P(y_i = NN \,, y_{i+1} = VB \,| x_1, \cdots, x_n)$$

$$c(NN \rightarrow apple) = \sum_i P(y_i = NN \,, x_i = apple | x_1, \cdots, x_n)$$

    - These are for one sentence. Take sum if multiple sentences.

These statistics summarize $P(\, y_1 \cdots y_{n+1} | x_1 \cdots x_n)$

Compute expected counts

$$c(NN) = \sum_i P(y_i = NN \,|x_1, \cdots, x_n)$$

$$= \sum_i \frac{P(x_1 \cdots x_n, y_i = NN)}{P(x_1 \cdots x_n)}$$

$$= \frac{\sum_i \alpha(i, y_i = NN)\beta(i, y_i = NN)}{\alpha(n+1, STOP)}$$

$$c(NN \rightarrow VB) = \sum_i P(y_i = NN, y_{i+1} = VB \,|x_1, \cdots, x_n)$$

$$= \sum_i \frac{P(x_1 \cdots x_n, y_i = NN, y_{i+1} = VB)}{P(x_1 \cdots x_n)}$$

$$= \frac{\sum_i \alpha(i, y_i = NN)\; q(VB|NN)\; e(x_{i+1}|VB)\; \beta(i+1, y_{i+1} = VB)}{\alpha(n+1, STOP)}$$

EM for HMM (Baum-Welch Algorithm)

- Initialize transition and emission parameters
  - Random, uniform, or more informed initialization
- Iterate until convergence
  - E-Step:
    - Compute expected counts
  - M-step:
    - Compute new parameter values to maximize expected log likelihood

$$\mathrm{E}_{Q(y_1 \cdots y_{n+1})}[\log P(x_1, \cdots, x_n, y_1 \cdots y_{n+1})]$$

    - Closed form solution: normalizing expected counts

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

These statistics summarize $P(y_1 \cdots y_{n+1} | x_1 \cdots x_n)$

HMM Unsupervised Learning

- Learn HMM given unannotated sequence $\{x_1, \cdots, x_n\}$
- Maximize marginal likelihood

$$P(x_1 \cdots x_n) = \sum_{y_1 \cdots y_n} P(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

- EM for HMM (Baum-Welch Algorithm)
- Can we directly optimize it by gradient descent?
  - Yes!
  - Use forward to compute $P(x_1 \cdots x_n)$
  - Run backprop on the computation graph

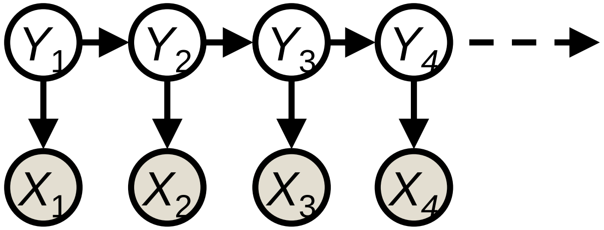
Forward-Backward is just backprop!

- The forward and then backprop procedure is almost the same as Forward-Backward
- Expected counts can be computed by backprop
  - $c(NN \to VB) = \frac{\partial \log P(x_1 \cdots x_n)}{\partial q(NN \to VB)}$
  - $c(NN \to apples) = \frac{\partial \log P(x_1 \cdots x_n)}{\partial e(NN \to apples)}$
- See https://aclanthology.org/W16-5901.pdf

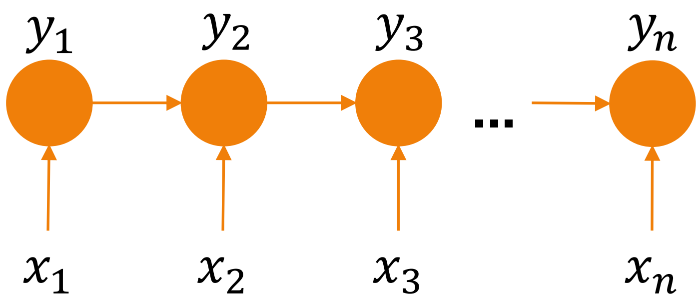
From HMM to Conditional Random Field

Beyond HMM

- The simplest method: for each word, predict its most frequent label
  - Problems:
    1. It does not consider the contextual info
    2. It does not consider relations between adjacent labels
- Does HMM solve the two problems?
  - HMM handles problem 2, but not 1



Max-Entropy Markov Models (MEMM)

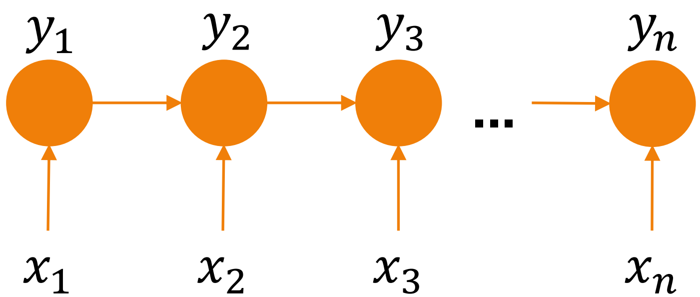


$$P(y_{1:n}|x_{1:n}) = P(y_1|x_1)\prod_{t=2}^{n} P(y_t|y_{t-1}, x_t)$$

$$P(y_t|y_{t-1}, x_t) = \frac{\exp(\boxed{\mathrm{s}(y_{t-1}, y_t, x_t)})}{Z(y_{t-1}, x_t)}$$

- Score function $s(y_{t-1}, y_t, x_t)$ can be a simple linear function $W^T f(y_{t-1}, y_t, x_t)$.
  - Possible features:
    - y_{t-1} is B and y_t is E?
    - y_{t-1} is B and y_t is O?
    - x_t is a noun?
    - x_t is capitalized? …

Max-Entropy Markov Models (MEMM)

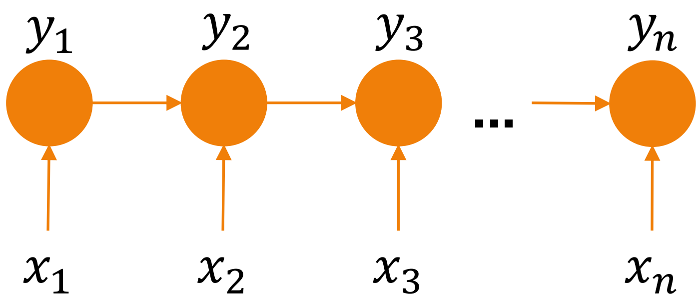


$$P(y_{1:n}|x_{1:n}) = P(y_1|x_1)\prod_{t=2}^{n} P(y_t|y_{t-1}, x_t)$$

$$P(y_t|y_{t-1}, x_t) = \frac{\exp(\boxed{s(y_{t-1}, y_t, x_t)})}{Z(y_{t-1}, x_t)}$$

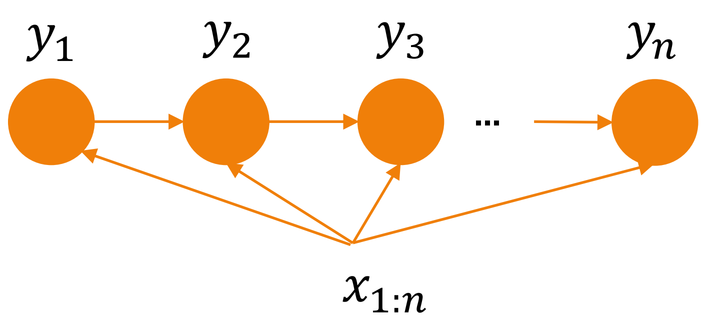
- Score function $s(y_{t-1}, y_t, x_t)$ can be a simple linear function $W^T f(y_{t-1}, y_t, x_t)$.
- It may also be a neural network with word embedding of x_t and label embedding of y_t and y_{t-1} as input
  - more on this later…
- Sometimes, $s(y_{t-1}, y_t, x_t)$ is decomposed to a transition score and an emission score
  - $s(y_{t-1}, y_t, x_t) = s_e(y_t, x_t) + s_q(y_t, y_{t-1})$

Max-Entropy Markov Models (MEMM)



$$P(y_{1:n}|x_{1:n}) = P(y_1|x_1)\prod_{t=2}^{n} P(y_t|y_{t-1}, x_t)$$

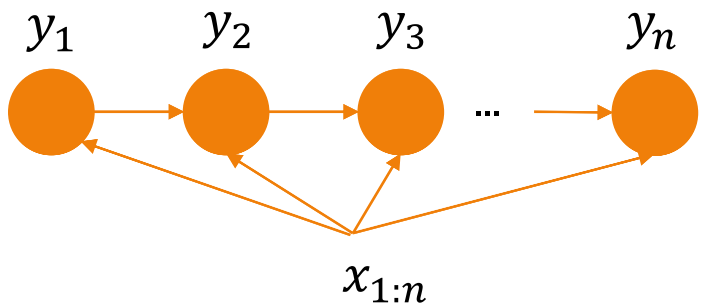
$$P(y_t|y_{t-1}, x_t) = \frac{\exp(s(y_{t-1}, y_t, x_t))}{Z(y_{t-1}, x_t)}$$



$$P(y_{1:n}|x_{1:n}) = P(y_1|x_{1:n})\prod_{t=2}^{n} P(y_t|y_{t-1}, x_{1:n})$$

$$P(y_t|y_{t-1}, x_{1:n}) = \frac{\exp(s(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$$

Max-Entropy Markov Models (MEMM)

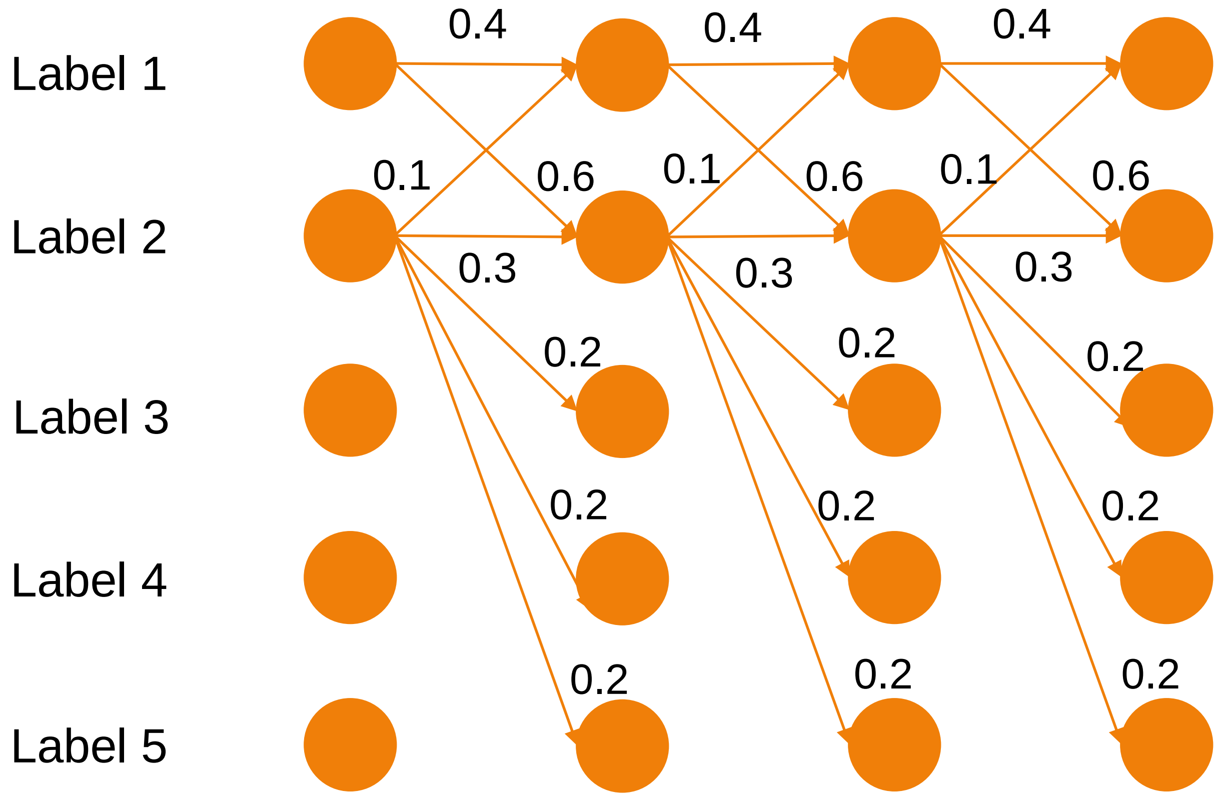


$$P(y_{1:n}|x_{1:n}) = P(y_1|x_{1:n}) \prod_{t=2}^{n} P(y_t|y_{t-1}, x_{1:n})$$

$$P(y_t|y_{t-1}, x_{1:n}) = \frac{\exp(s(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$$

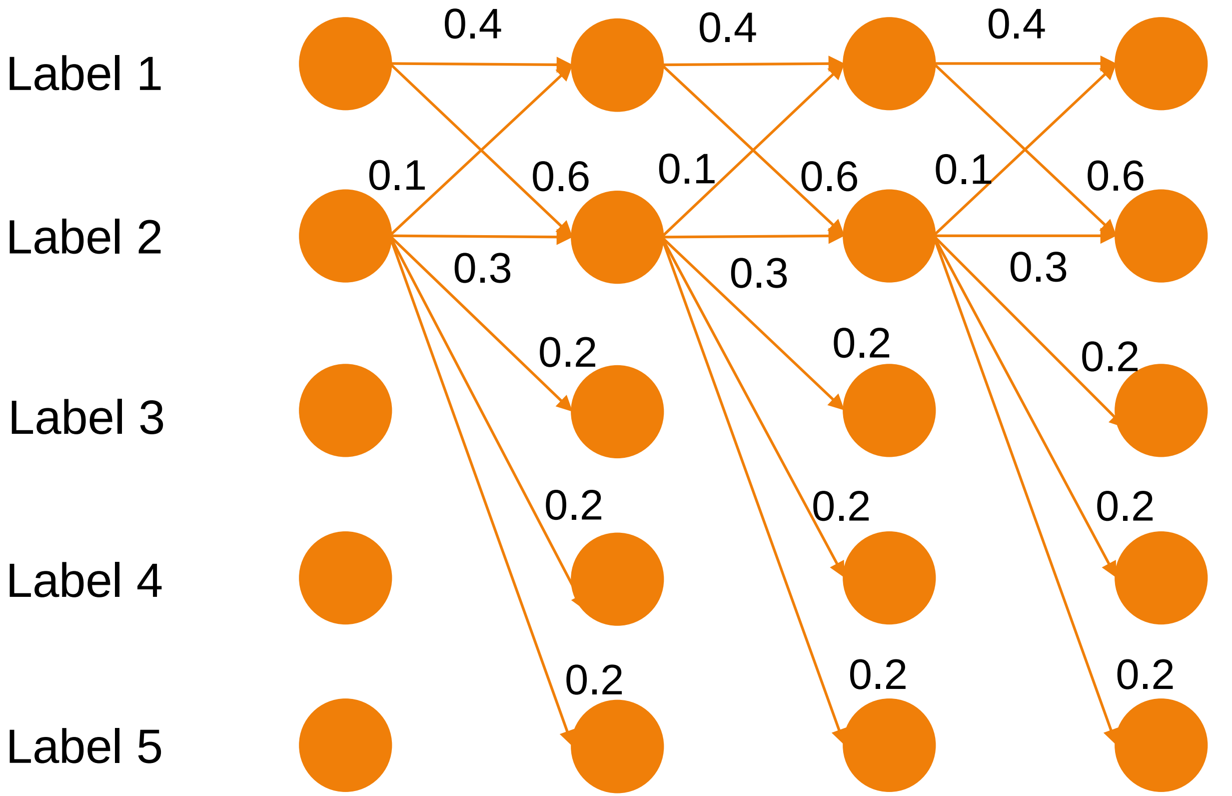
- Now we can consider info from the whole sentence in the score function
- MEMM considers both contextual info and relations between adjacent labels!
- But… MEMM suffers from label bias problem

Label Bias Problem



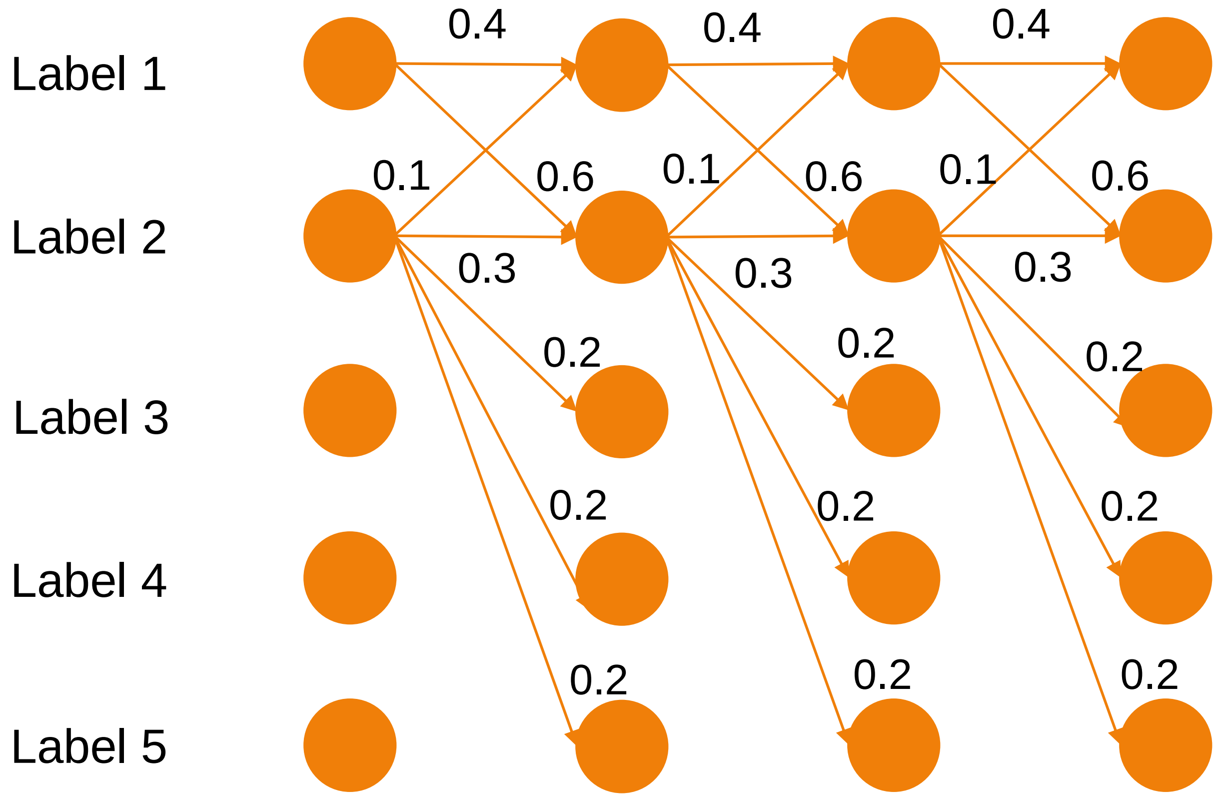
- What the local transition probabilities say:
  - Label 1 prefers to go to label 2
  - Label 2 prefers to stay at label 2

Label Bias Problem



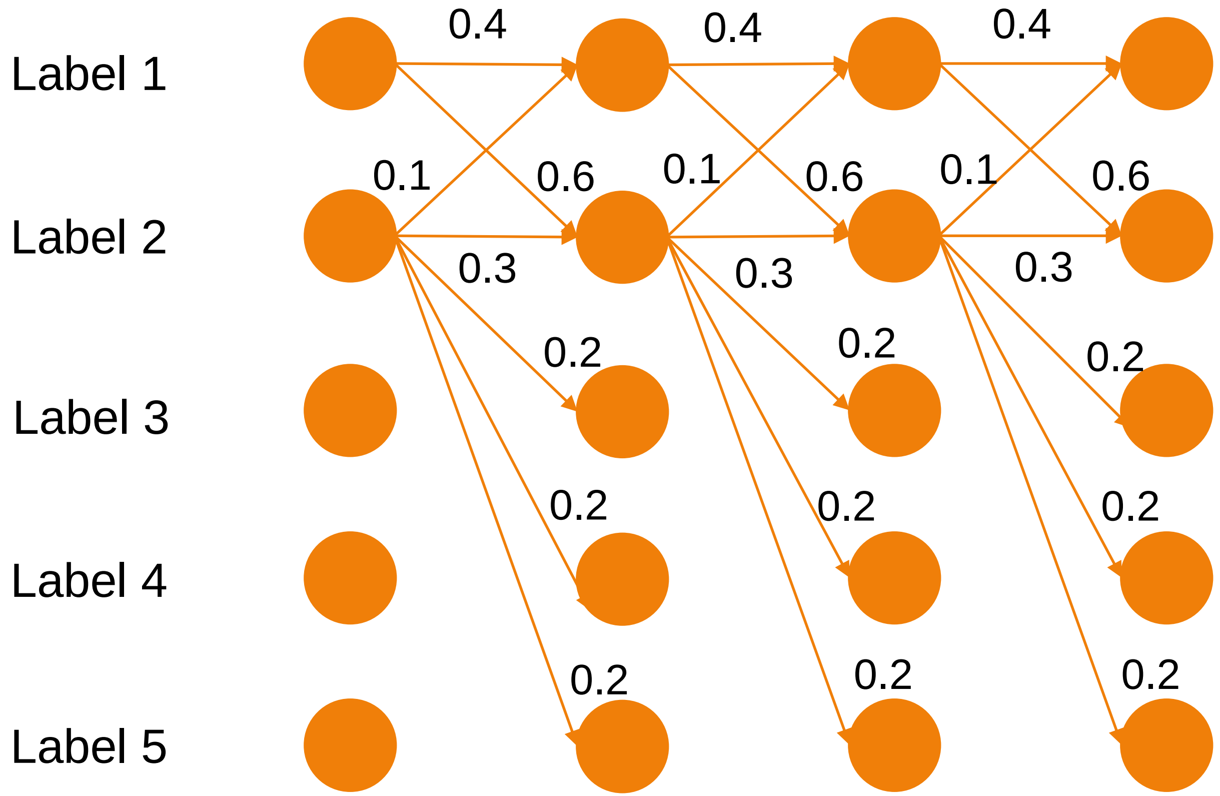
- $\boxed{P(1 \rightarrow 1 \rightarrow 1 \rightarrow 1)=0.4^3=0.064}$
- $P(1 \rightarrow 2 \rightarrow 1 \rightarrow 2)=0.6*0.1*0.6 =0.036$
- $P(2 \rightarrow 2 \rightarrow 2 \rightarrow 2)=0.3^3=0.027$
- $P(2 \rightarrow 1 \rightarrow 2 \rightarrow 1)=0.1*0.6*0.1 =0.006$

Label Bias Problem



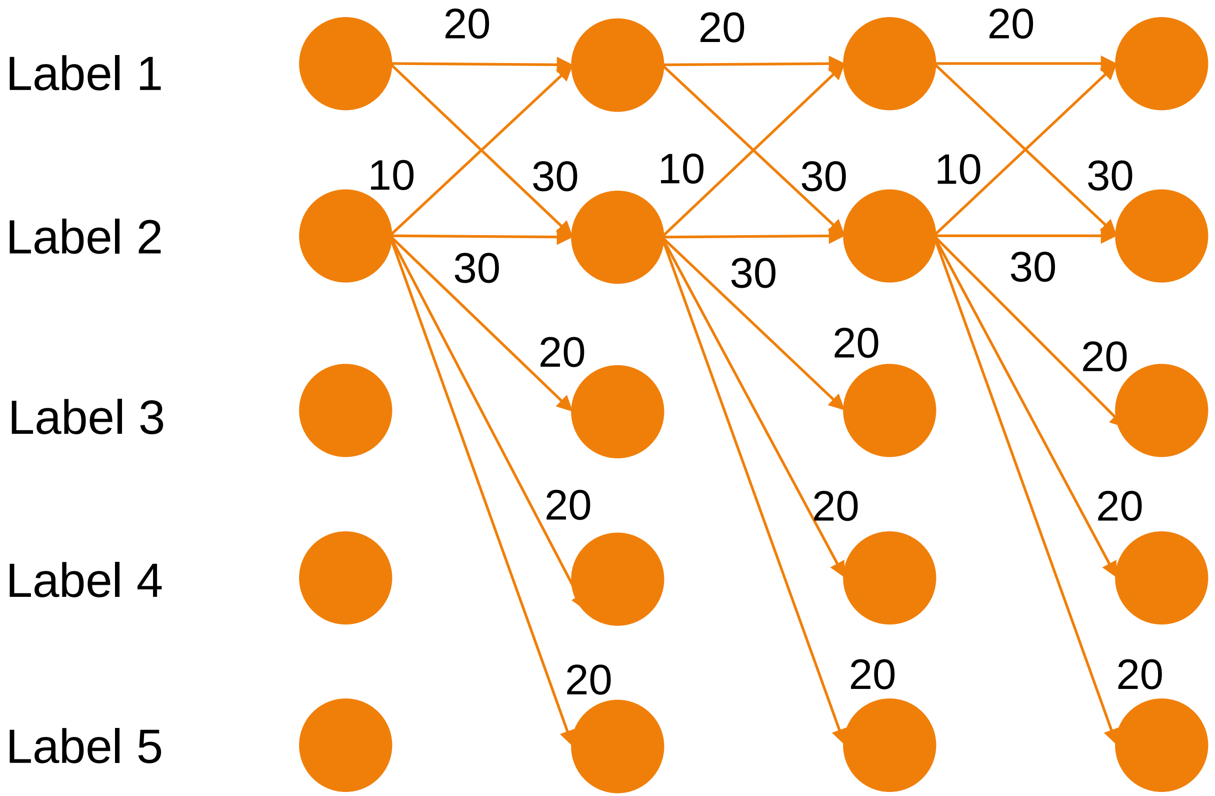
- Label 1 has only two transitions but label 2 has five
- Transition probabilities from label 2 are lower

Label Bias Problem



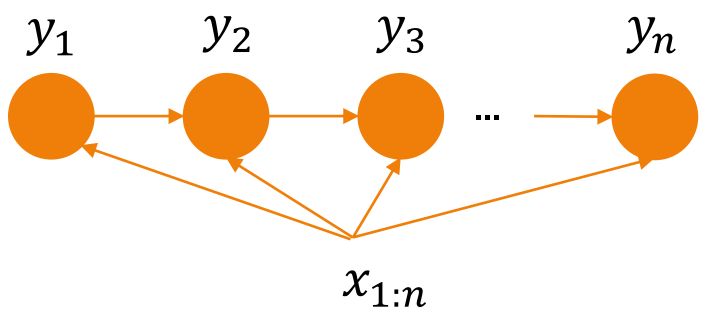
- Label bias in MEMM
  - Preference of states with lower number of transitions

Label Bias Problem



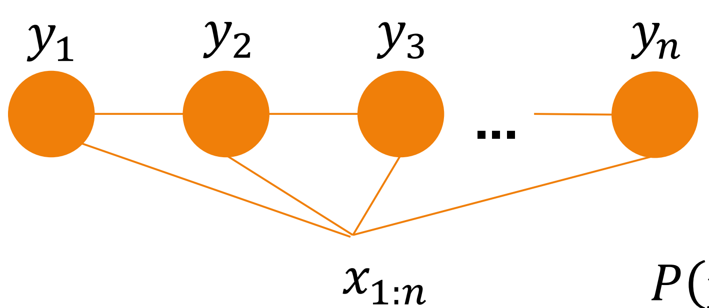
- Solution
  - From local probabilities to local potentials

From MEMM to CRF



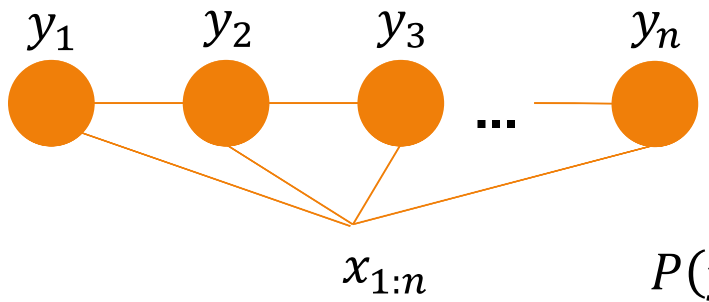
$$P(y_{1:n}|x_{1:n}) = P(y_1|x_{1:n}) \prod_{t=2}^{n} P(y_t|y_{t-1}, x_{1:n})$$

$$P(y_t|y_{t-1}, x_{1:n}) = \frac{\exp(s(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$$



$$P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{t=1}^{n} \exp(s(y_{t-1}, y_t, x_{1:n}))$$

From MEMM to CRF



$$P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})}\prod_{t=1}^{n}\exp(s(y_{t-1}, y_t, x_{1:n}))$$

- Conditional Random Field (CRF) is an undirected graphical model
  - Global normalization instead of local normalization
  - Both problems solved ✓
  - Label bias solved ✓

CRF inference (decoding)

$$y^* = \underset{y_1 \cdots y_n}{\operatorname{argmax}} \frac{1}{Z(x_{1:n})} \prod_{t=1}^{n} \exp(s(y_{t-1}, y_t, x_{1:n}))$$

$$= \underset{y_1 \cdots y_n}{\operatorname{argmax}} \prod_{t=1}^{n} \exp(s(y_{t-1}, y_t, x_{1:n}))$$

$$= \underset{y_1 \cdots y_n}{\operatorname{argmax}} \boxed{\sum_{t=1}^{n} s(y_{t-1}, y_t, x_{1:n})}$$

Score of label sequence
$s(y_{1:n})$

- Decoding by Viterbi

$$\boxed{\pi(0, \mathrm{START}) = 0}$$

$$\pi(i, y_i) = \max_{y_1 \cdots y_{i-1}} \sum_{t=1}^{i} s(y_{t-1}, y_t, x_{1:n})$$

$$= \max_{y_{i-1}} s(y_{i-1}, y_i, x_{1:n}) + \max_{y_1 \cdots y_{i-2}} \sum_{t=1}^{i-1} s(y_{t-1}, y_t, x_{1:n})$$

$$= \max_{y_{i-1}} s(y_{i-1}, y_i, x_{1:n}) + \pi(i-1, y_{i-1})$$

CRF Supervised Learning

- Learn CRF given annotated sequence $\{(x_1, y_1), \cdots, (x_n, y_n)\}$
- Maximizing conditional (log) likelihood

$$P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})} \exp\left(\sum_{t=1}^{n} s(y_{t-1}, y_t, x_{1:n})\right)$$

$$Z(x_{1:n}) = \sum_{y'} \exp\left(\sum_{t=1}^{n} s(y'_{t-1}, y'_t, x_{1:n})\right)$$

- Optimization with gradient descent
  - The partition function Z is computed by Forward algorithm
  - The gradient formula involves expected counts
    - Can be computed with Forward-Backward
    - Or we simply let auto-differentiation handle everything (as discussed earlier)

CRF Supervised Learning

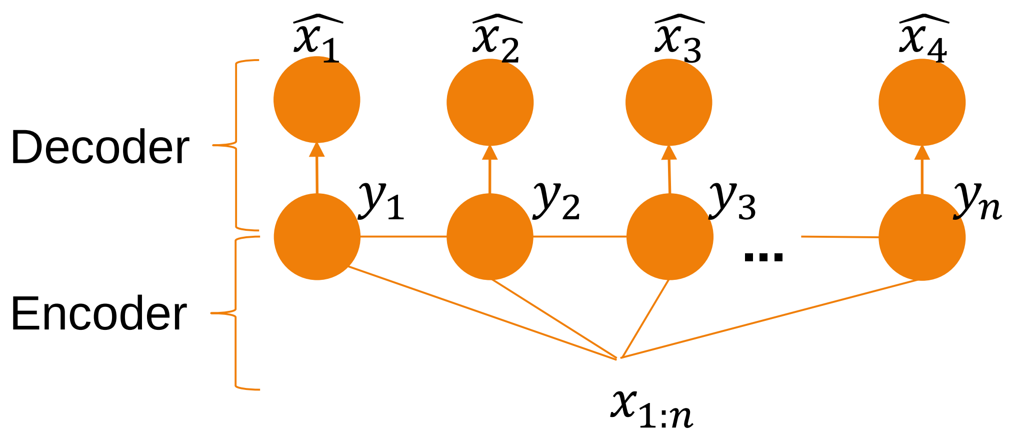
- Learn CRF given annotated sequence $\{(x_1, y_1), \cdots, (x_n, y_n)\}$
- Minimizing margin-based loss (Structured SVM)

$$L_{SSVM} = \max_{y_{1:n}}\left(s(y_{1:n}) + \Delta(y_{1:n}, y^\star_{1:n}) - s(y^\star_{1:n})\right)$$

  - $\Delta(y, y^\star) \geq 0$ is the cost we incur when we predict y but the truth is y*
  - $\max_{y_{1:n}}(\cdots)$ can be computed with Viterbi if Δ is position-wise decomposable, e.g., num of different labels
  - Advantages
    - take into account the Δ cost
    - focus on the decision boundary instead of the full distribution
- Optimization --- loss not differentiable
  - stochastic subgradient descent
  - quadratic programming (cutting-plane method)

CRF Unsupervised Learning

- Learn CRF given unannotated sequence $\{x_1, \cdots, x_n\}$
- Impossible to compute $P(x_1, \cdots, x_n)$ with a CRF!
- CRF autoencoder (CRF-AE)
  - Encoder: CRF
  - Decoder: simply predict each word from its tag



CRF Unsupervised Learning

- Learn CRF given unannotated sequence $\{x_1, \cdots, x_n\}$
- Impossible to compute $P(x_1, \cdots, x_n)$ with a CRF!
- CRF autoencoder (CRF-AE)
  - Encoder: CRF
  - Decoder: simply predict each word from its tag
- Training loss:

$$P(\widehat{x_{1:n}}\ |x_{1:n}) = \sum_{y_{1:n}} P(y_{1:n}|x_{1:n}) P(\widehat{x_{1:n}}\ |y_{1:n})$$

$$= \sum_{y_{1:n}} \frac{1}{Z(x_{1:n})} \prod_{t=1}^{n} \exp\left(s(y_{t-1}, y_t, x_{1:n})\right) P(\widehat{x_i}\ |y_i)$$

  - The loss can be computed with Forward algorithm and optimized with gradient descent

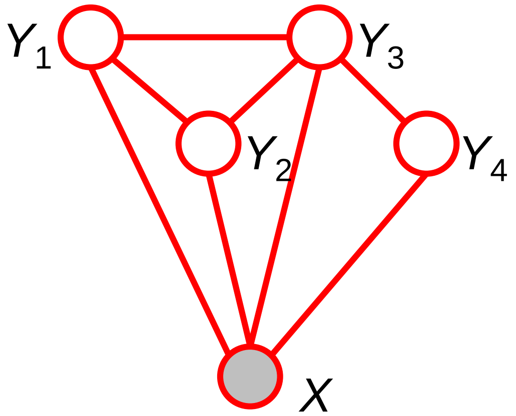
CRF in general

- An extension of Markov networks (aka. Markov random fields) where everything is conditioned on the input

$$P(y|x) = \frac{1}{Z(x)} \prod_{C} \psi_C(y_C, x)$$

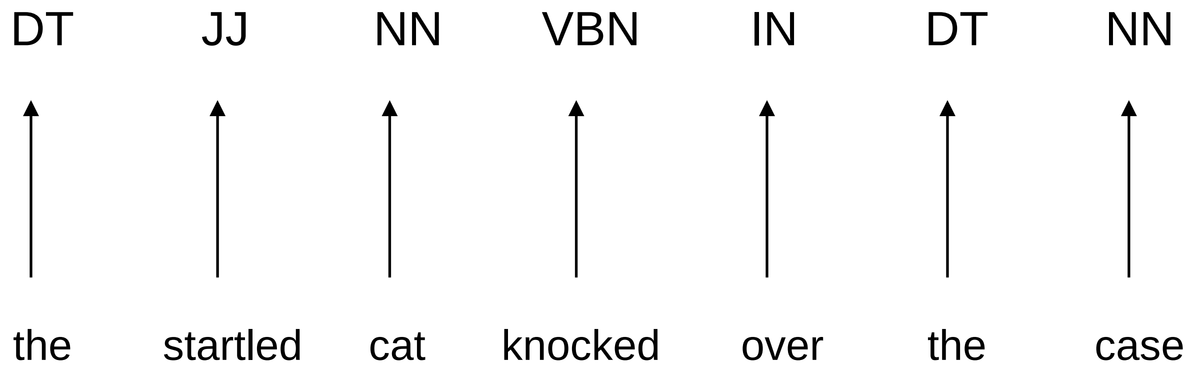
  - where $\psi_C(y_C, x)$ is the potential over clique C and $Z(x)$ is the normalization coefficient.

$$Z(x) = \sum_{y} \prod_{C} \psi_C(y_C, x)$$



Neural Sequence Labeling Model

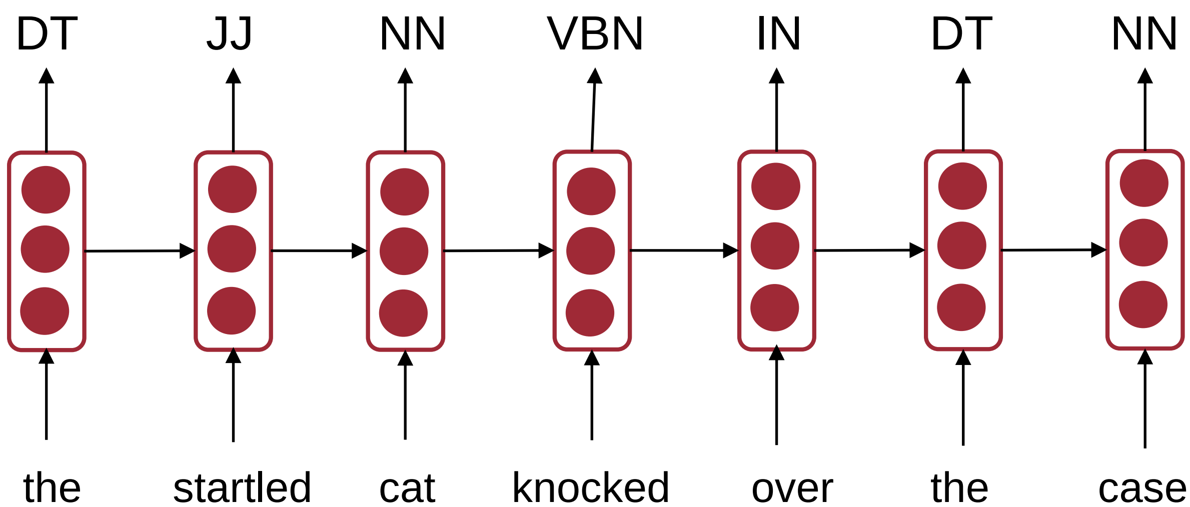
Simplest neural method



- Predicting labels directly from static word embeddings
  - Problem 1: it does not utilize the context of each word
  - Problem 2: it does not utilize relations between neighboring labels

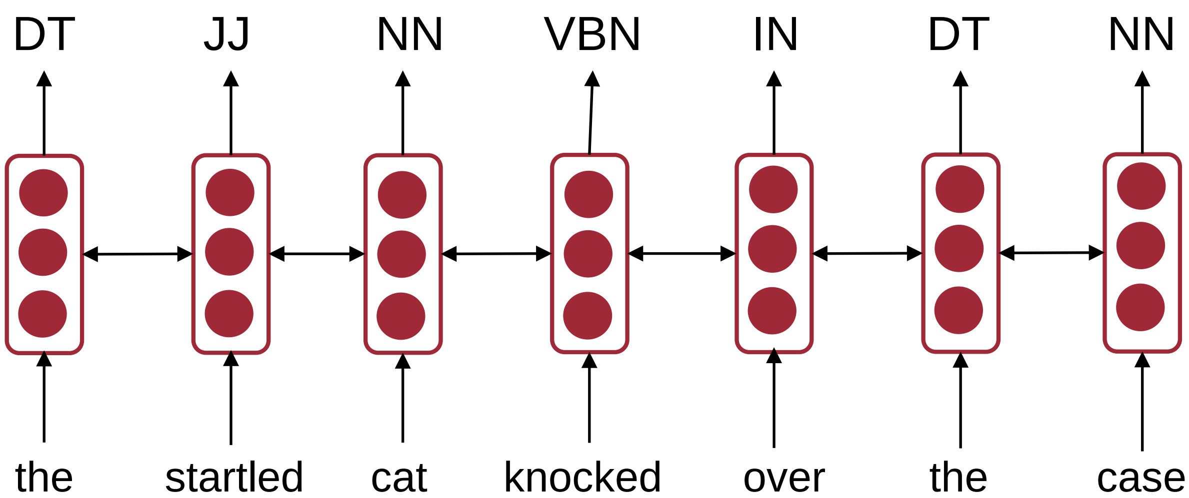
RNN for sequence labeling

RNNs in the general sense, including LSTMs and GRUs



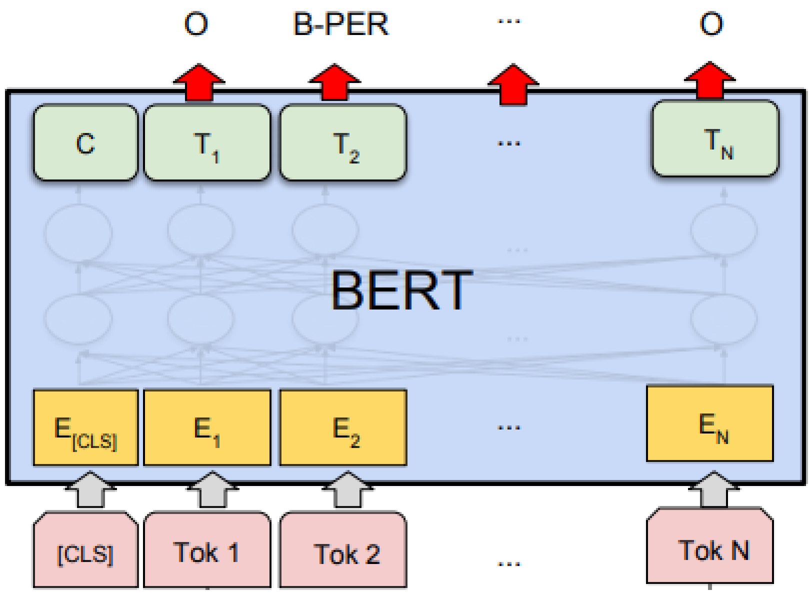
- Predicting labels from RNN hidden vectors
  - Problem 1: it does not utilize the context of each word
    - Each hidden vector only incorporates info from the left context
  - Problem 2: it does not utilize relations between neighboring labels

Bidirectional RNN



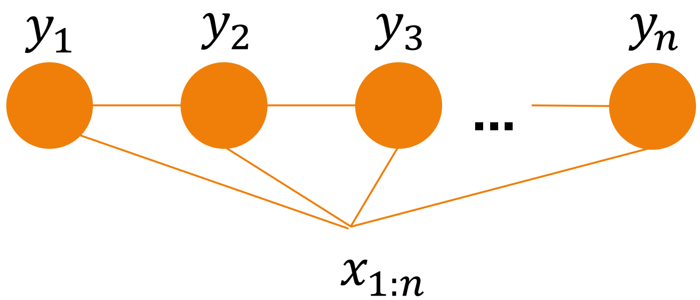
- Predicting labels from bi-RNN hidden vectors
  - Problem 1: it does not utilize the context of each word
    - Solved!
  - Problem 2: it does not utilize relations between neighboring labels

Transformer



- Predicting labels from Transformer output vectors
  - Problem 1: it does not utilize the context of each word
    - Solved!
  - Problem 2: it does not utilize relations between neighboring labels

Neural CRF



$$P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{t=1}^{n} \exp(s(y_{t-1}, y_t, x_{1:n}))$$

- Use a neural model (RNN, Transformer, or both) to compute CRF potentials (typically only the emission scores)
  - Both problems solved!
  - The default model for sequence labeling nowadays

Inference and Learning

- For *all* these models:
  - Inference
    - Without CRF: independent prediction at each position
      - Sometimes called neural softmax
    - With CRF: Viterbi
  - Learning
    - Optimize conditional log likelihood or margin-based loss
    - Similar to those in CRF learning

Summary

Sequence Labeling

- Hidden Markov model (HMM)
  - Inference: Viterbi, Forward, Backward
  - Learning: Maximum Likelihood Estimate, Expectation-Maximization / SGD
- Conditional random filed (CRF)
  - Label bias problem
  - Inference: Viterbi, Forward, Backward
  - Learning: conditional likelihood, margin-based loss, CRF-AE
- Neural models
  - Neural softmax, neural CRF