CS6741: Structured Prediction for NLP
Fall 2015

Hidden Markov Models

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Luke Zettlemoyer, Yejin Choi, Chris Manning, and Dan Jurafsky

Overview

- Hidden Markov Models
- Learning
  - Supervised: Maximum Likelihood
- Inference (or Decoding)
  - Viterbi
  - Forward Backward

Sequence-to-Sequence

Consider the problem of jointly modeling a pair of strings

– E.g.: part of speech tagging

DT	NNP	NN	VBD	VBN	RP	NN	NNS
The	Georgia	branch	had	taken	on	loan	commitments …
DT	NN	IN	NN		VBD	NNS	VBD
The	average	of	interbank		offered	rates	plummeted …

Q: How do we map each word in the input sentence onto the appropriate label?

A: We can learn a joint distribution:

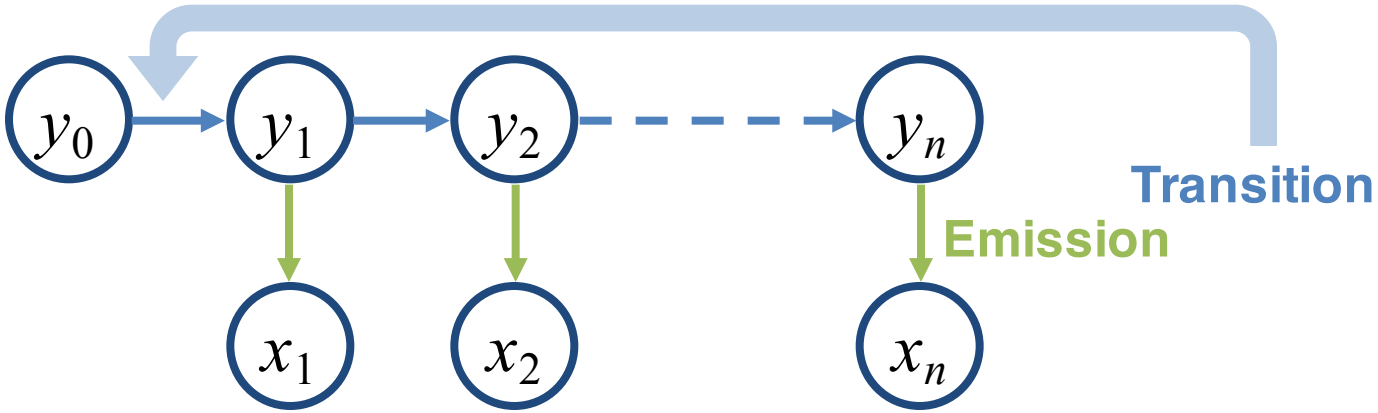
$$p(x_1 \dots x_n, y_1 \dots y_n)$$

And then compute the most likely assignment:

$$\arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

Classic Solution: HMMs

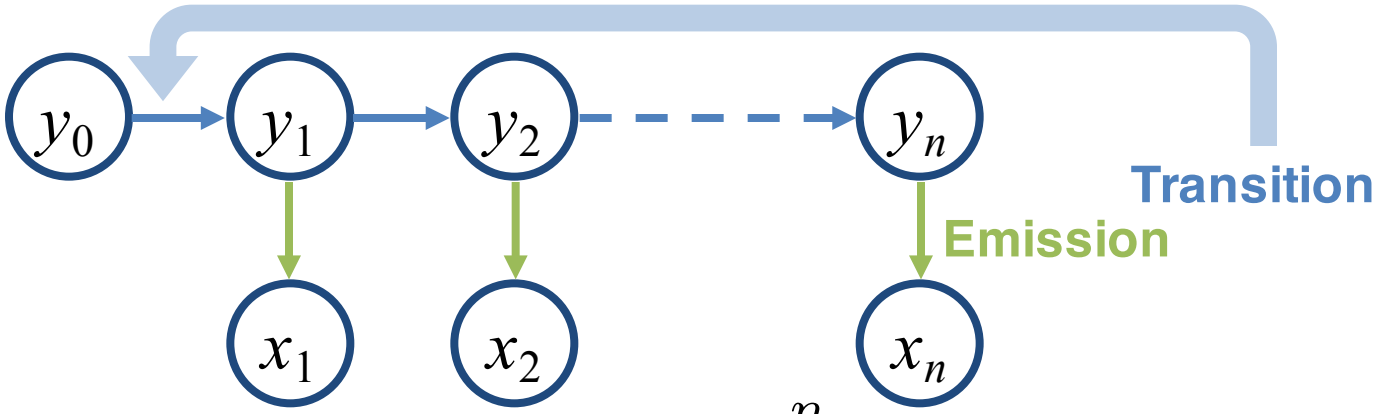
We want a model of sequences y and observations x



$$p(x_1 \ldots x_n, y_1 \ldots y_n) =$$

where y_0=*START* and we call q(y'|y) the **transition** distribution and e(x|y) the **emission** (or observation) distribution.

Model Assumptions



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

- 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?

HMM for POS Tagging

The Georgia branch had taken on loan commitments …

DT NNP NN VBD VBN RP NN NNS

- HMM Model:
  - States Y =
  - Observations X =
  - Transition dist'n q(yi |yi -1) models
  - Emission dist'n e(xi |yi) models

HMM Inference and Learning

- Learning
  - Maximum likelihood: transitions q and emissions e

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

- Inference
  - Viterbi

$$y^* = \arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

  - Forward backward

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

Learning: Maximum Likelihood

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

- Maximum likelihood methods for estimating transitions q and emissions e

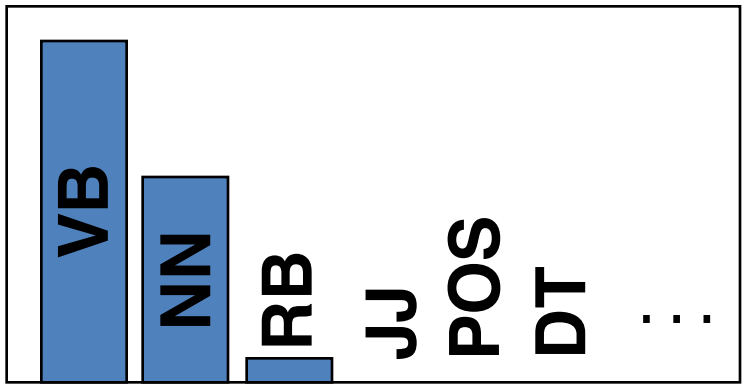
$$q_{ML}(y_i|y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \qquad e_{ML}(x|y) = \frac{c(y, x)}{c(y)}$$

- Will these estimates be high quality?
  - Which is likely to be more sparse, q or e?
- Smoothing?

Smoothing Review

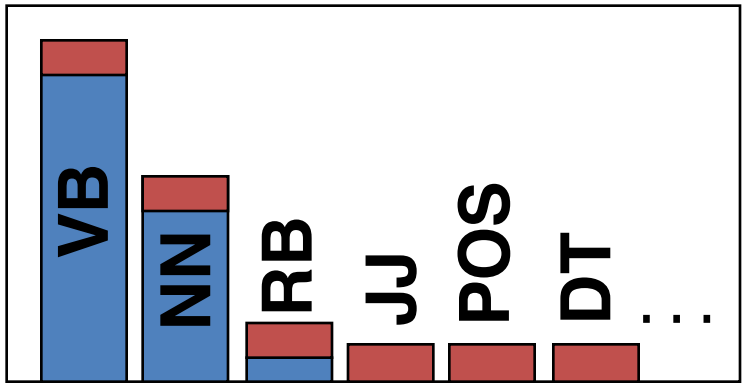
- Sparsity means that we often get spikey distributions from data:

P(w | “back”)
20 VB
10 NN
1 RB
0 JJ



- Smoothing flattens spiky distributions so they generalize better

P(w | “back”)
21 VB
11 NN
2 RB
1 JJ



- Very important all over NLP (and ML more generally), but easy to do badly!
- Question: what is the best way to do it?

Learning: Low Frequency Words

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

- Typically, for transitions:
  - Linear Interpolation

$$q(y_i|y_{i-1}) = \lambda_1 q_{ML}(y_i|y_{i-1}) + \lambda_2 q_{ML}(y_i)$$

- However, other approaches used for emissions
  - **Step 1:** Split the vocabulary
    - Frequent words: appear more than M (often 5) times
    - Low frequency: everything else
  - **Step 2:** Map each low frequency word to one of a small, finite set of possibilities
    - For example, based on prefixes, suffixes, etc.
  - **Step 3:** Learn model for this new space of possible word sequences

Another Example: Chunking

- Goal: Segment text into spans with certain properties
- For example, named entities: PER, ORG, and LOC

Germany 's representative to the European Union 's veterinary committee Werner Zwingman said on Wednesday consumers should…

[Germany]$_{LOC}$'s representative to the [European Union]$_{ORG}$'s veterinary committee [Werner Zwingman]$_{PER}$ said on Wednesday consumers should…

How is this a sequence tagging problem?

Another Example: Chunking

Germany 's representative to the European Union 's veterinary committee Werner Zwingman said on Wednesday consumers should…

$[\text{Germany}]_{\text{LOC}}$'s representative to the $[\text{European Union}]_{\text{ORG}}$'s veterinary committee $[\text{Werner Zwingman}]_{\text{PER}}$ said on Wednesday consumers should…

- HMM Model:
  - States Y = {NA,BL,CL,BO,CO,BP,CP} represent beginnings (BL,BO,BP) and continuations (CL,CO,CP) of chunks, as well as other words (NA)
  - Observations X = V are words
  - Transition dist'n q(yi |yi -1) models the tag sequences
  - Emission dist'n e(xi |yi) models words given their type

Low Frequency Words: An Example

- Named Entity Recognition [Bickel et. al, 1999]
  - Used the following word classes for infrequent words:

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

Low Frequency Words: An Example

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

firstword/NA soared/NA at/NA initCap/SC Co./CC ,/NA easily/NA lowercase/NA forecasts/NA on/NA initCap/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP initCap/CP announced/NA first/NA quarter/NA results/NA ./NA

- NA = No entity
- SC = Start Company
- CC = Continue Company
- SL = Start Location
- CL = Continue Location
- …

HMM Inference and Learning

- Learning
  - Maximum likelihood: transitions q and emissions e

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

- Inference
  - Viterbi

$$y^* = \arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

  - Forward backward

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

Inference (Decoding)

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

$$y^* = \arg\max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

- Given model parameters, we can score any sequence pair

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

q(NNP|♦) e(Fed|NNP) q(VBZ|NNP) e(raises|VBZ) q(NN|VBZ).....

- In principle, we're done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

NNP VBZ NN NNS CD NN . → *logP = -23*

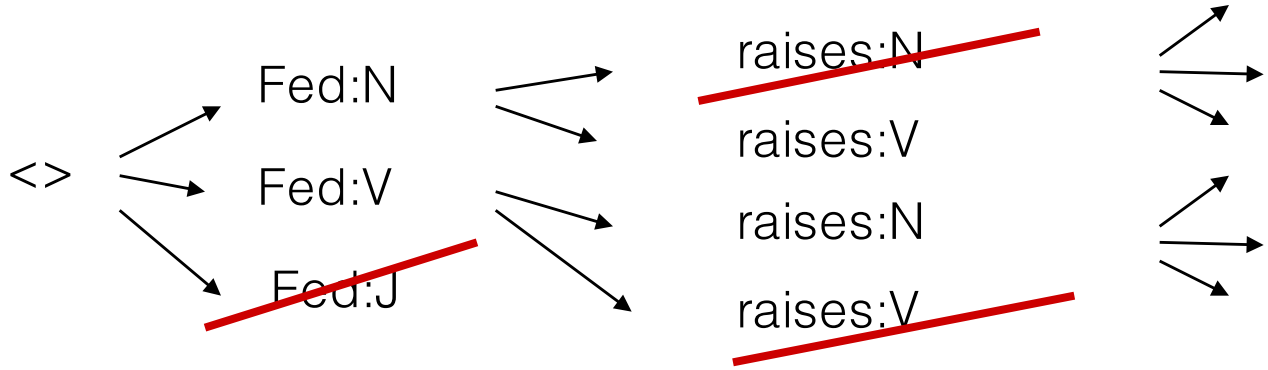
NNP NNS NN NNS CD NN . → *logP = -29*

NNP VBZ VB NNS CD NN . → *logP = -27*

Any issue?

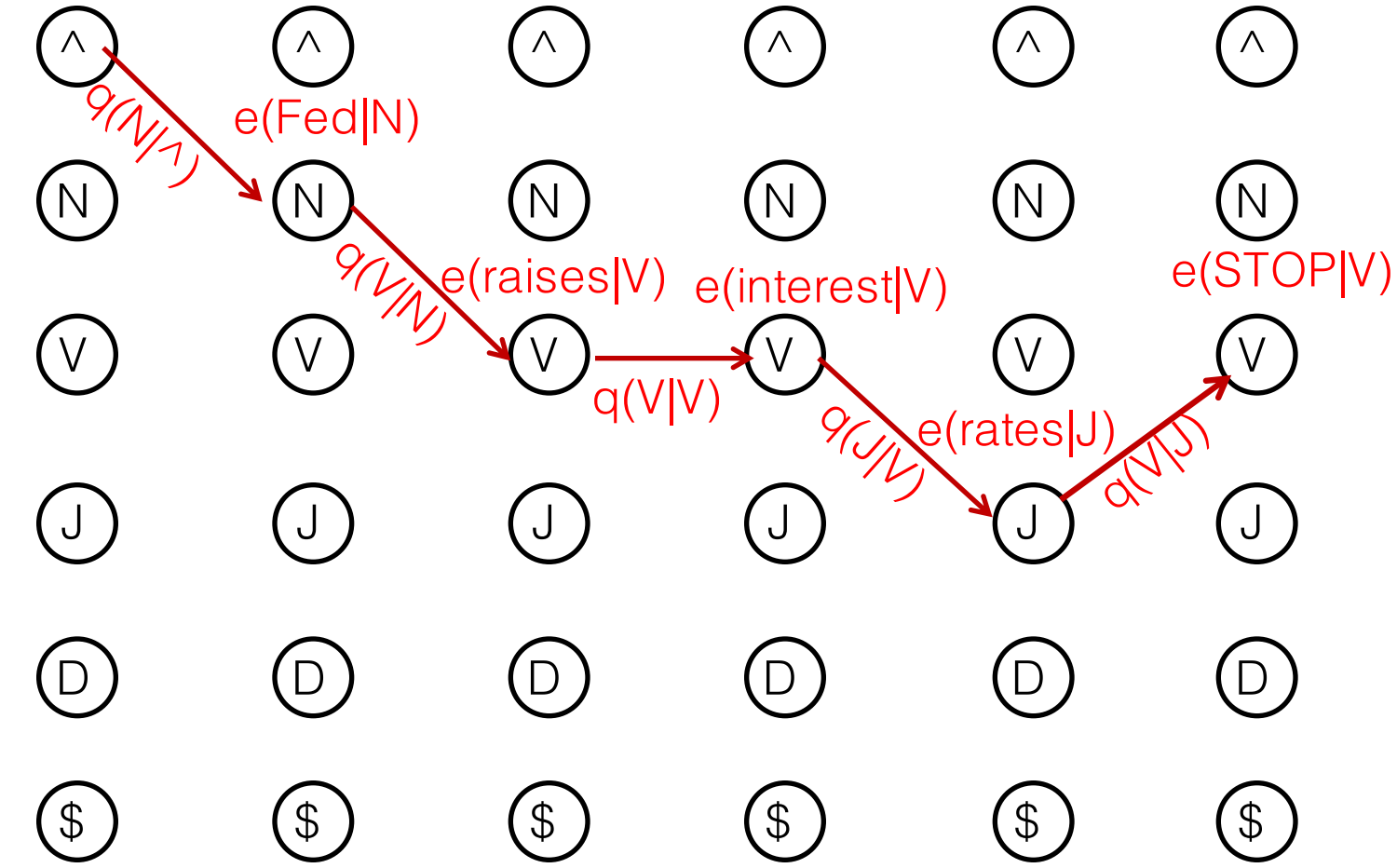
Finding the Best Trajectory

- Too many trajectories (state sequences) to list
- Option 1: Beam Search
  - A beam is a set of partial hypotheses
  - Start with just the single empty trajectory
  - At each derivation step:
    - Consider all continuations of previous hypotheses
    - Discard most, keep top k



- Beam search often works OK in practice, but …
  - … but sometimes you want the optimal answer
  - … and there's usually a better option than naïve beams

The State Lattice / Trellis



START ^ | Fed N | raises V | interest V | rates J | STOP V

Scoring a Sequence

$$y^* = \arg\max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

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

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

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

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

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

- We can now design an efficient algorithm.
  - How?

The Viterbi Algorithm

Dynamic program for computing (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:

// Store score

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

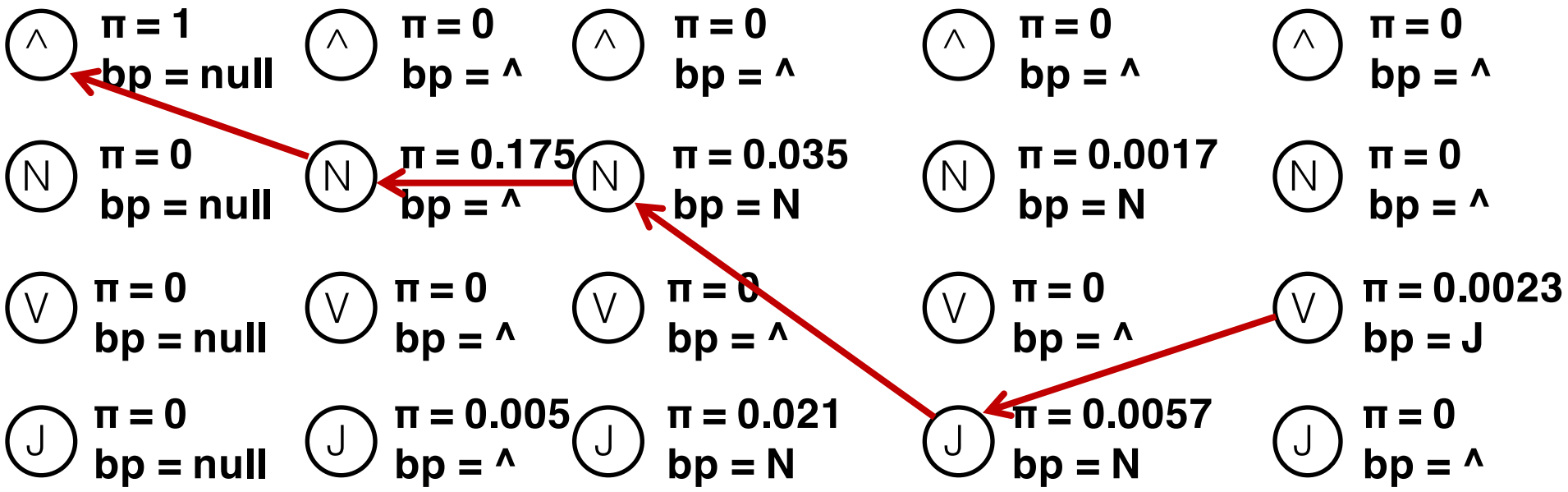
// Store back-pointer

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

What for?

Tie breaking:
Prefer first

The State Lattice / Trellis



from \ to	^	N	V	J
^	0.0	0.7	0.2	0.1
N	0.0	0.4	0.3	0.3
V	0.0	0.0	1.0	0.0
J	0.0	0.4	0.4	0.2

emissions	START	Fed	raises	interest	STOP
^	1.0	0.5	0.2	0.1	0.0
N	0.0	0.25	0.5	0.1	0.0
V	0.0	0.0	0.0	0.0	1.0
J	0.0	0.05	0.4	0.55	0.0

The Viterbi Algorithm: Runtime

- In term of sentence length n?
  - Linear
- In term of number of states |K|?
  - Polynomial

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

- 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: $O(n|\mathcal{K}|^2)$
- Q: Is this a practical algorithm?
- A: depends on |K|….

Tagsets in Different Languages

Language	Source	# Tags
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	[illegible]
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	[illegible]
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	[illegible]
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	[illegible]
German	Tiger/CoNLL06 (Brants et al., 2002)	54
German	Negra (Skut et al., 1997)	54
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38
Hungarian	Szeged/CoNLL07 (Csendes et al., 2005)	43
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42
Korean	Sejong (http://www.sejong.or.kr)	187
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	[illegible]
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	[illegible]
Spanish	Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)	47
Swedish	Talbanken05/CoNLL06 (Nivre et al., 2006)	41
Turkish	METU-Sabanci/CoNLL07 (Oflazer et al., 2003)	31

$294^2 = 86436$

$45^2 = 2045$

$11^2 = 121$

[Petrov et al. 2012]

HMM Inference and Learning

- Learning
  - Maximum likelihood: transitions q and emissions e

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

- Inference
  - Viterbi

$$y^* = \arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

  - Forward backward

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

Marginal Inference

- Problem: find the marginal probability of each tag for y_i

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

- Given model parameters, we can score any sequence pair

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

q(NNP|♦) e(Fed|NNP) q(VBZ|NNP) e(raises|VBZ) q(NN|VBZ).....

- In principle, we're done – list all possible tag sequences, score each one, **sum over all of the possible values for y_i**

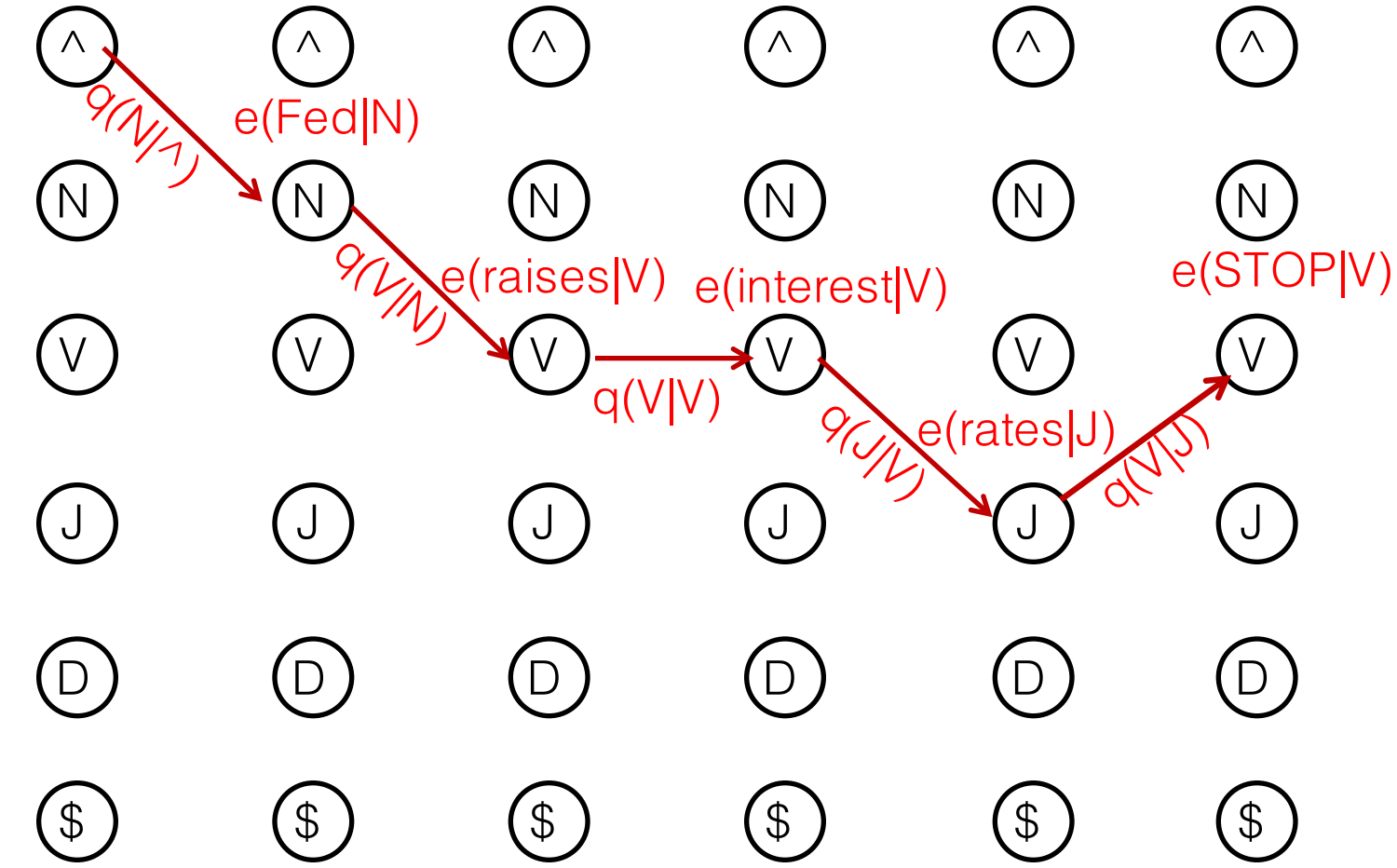
NNP VBZ NN NNS CD NN . → *logP = -23*

NNP NNS NN NNS CD NN . → *logP = -29*

NNP VBZ VB NNS CD NN . → *logP = -27*

Any issue?

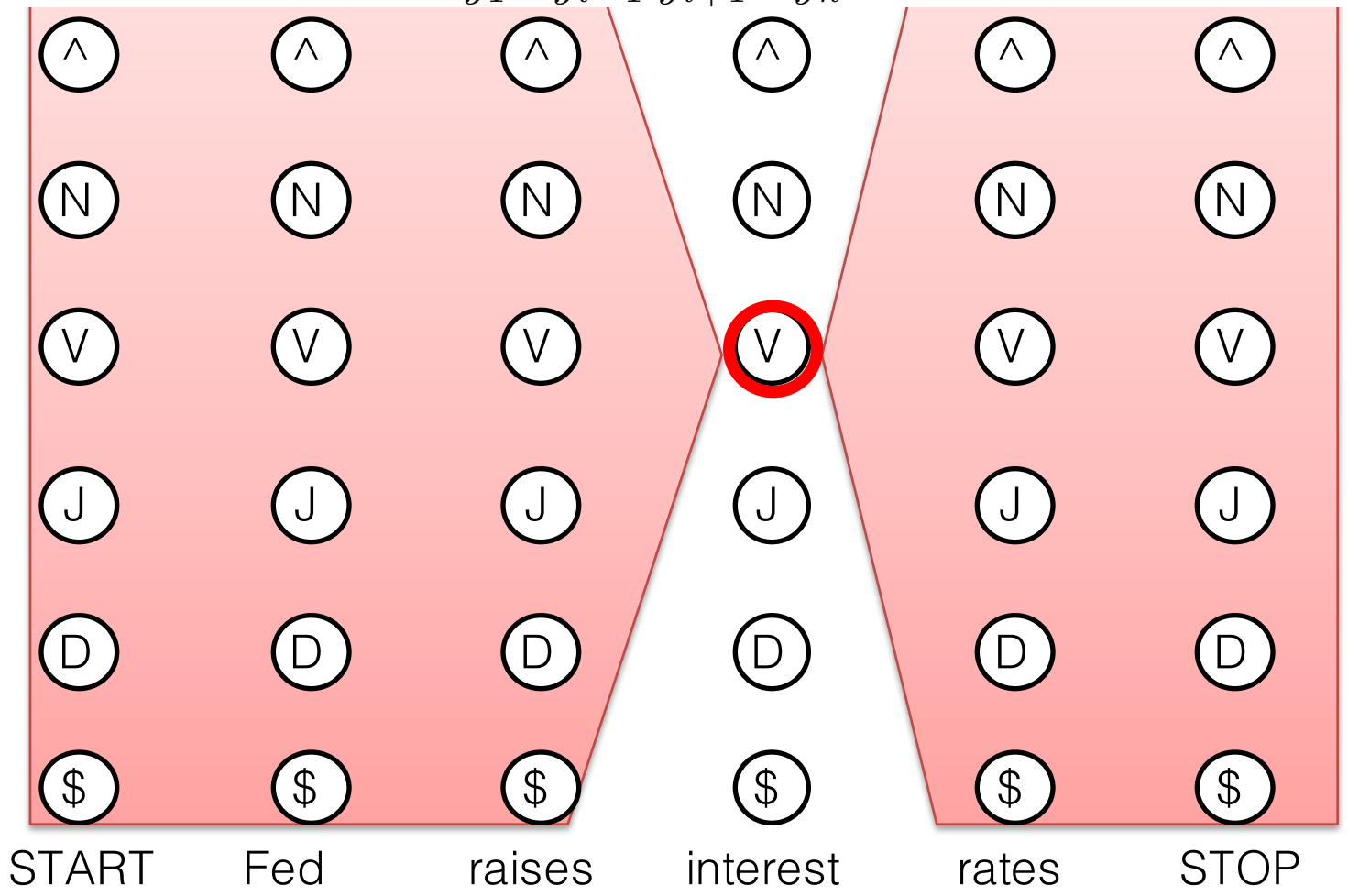
The State Lattice / Trellis



START ^ | Fed N | raises V | interest V | rates J | STOP V

The State Lattice / Trellis: Marginal

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



Marginal Probability

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

Sum over all paths, on both sides of each y_i

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

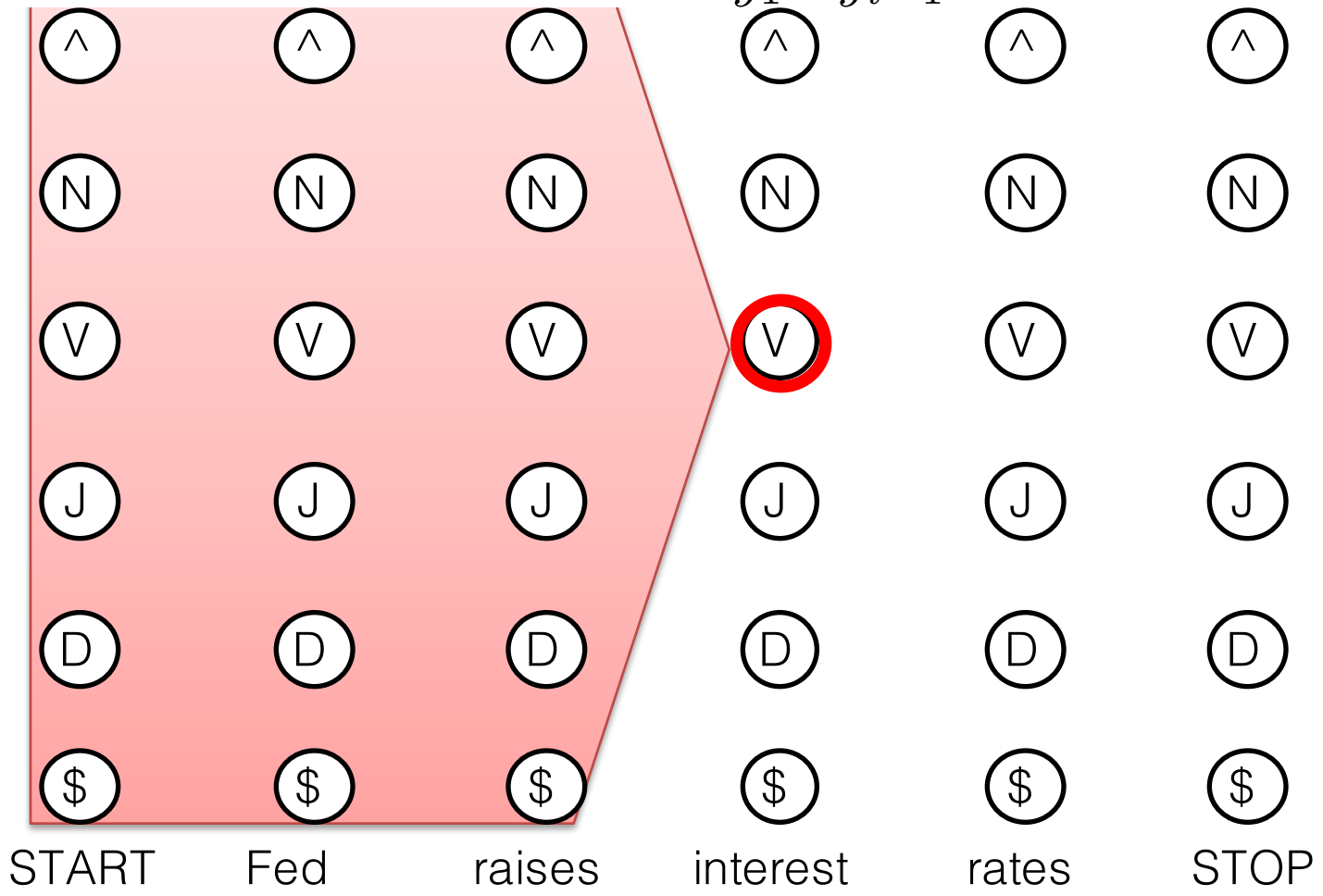
$$= \sum_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \alpha(i-1, y_{i-1})$$

$$\beta(i, y_i) = p(x_{i+1} \ldots x_n \mid y_i) = \sum_{y_{i+1} \ldots y_n} p(x_{i+1} \ldots x_n, y_{i+1} \ldots y_n \mid y_i)$$

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

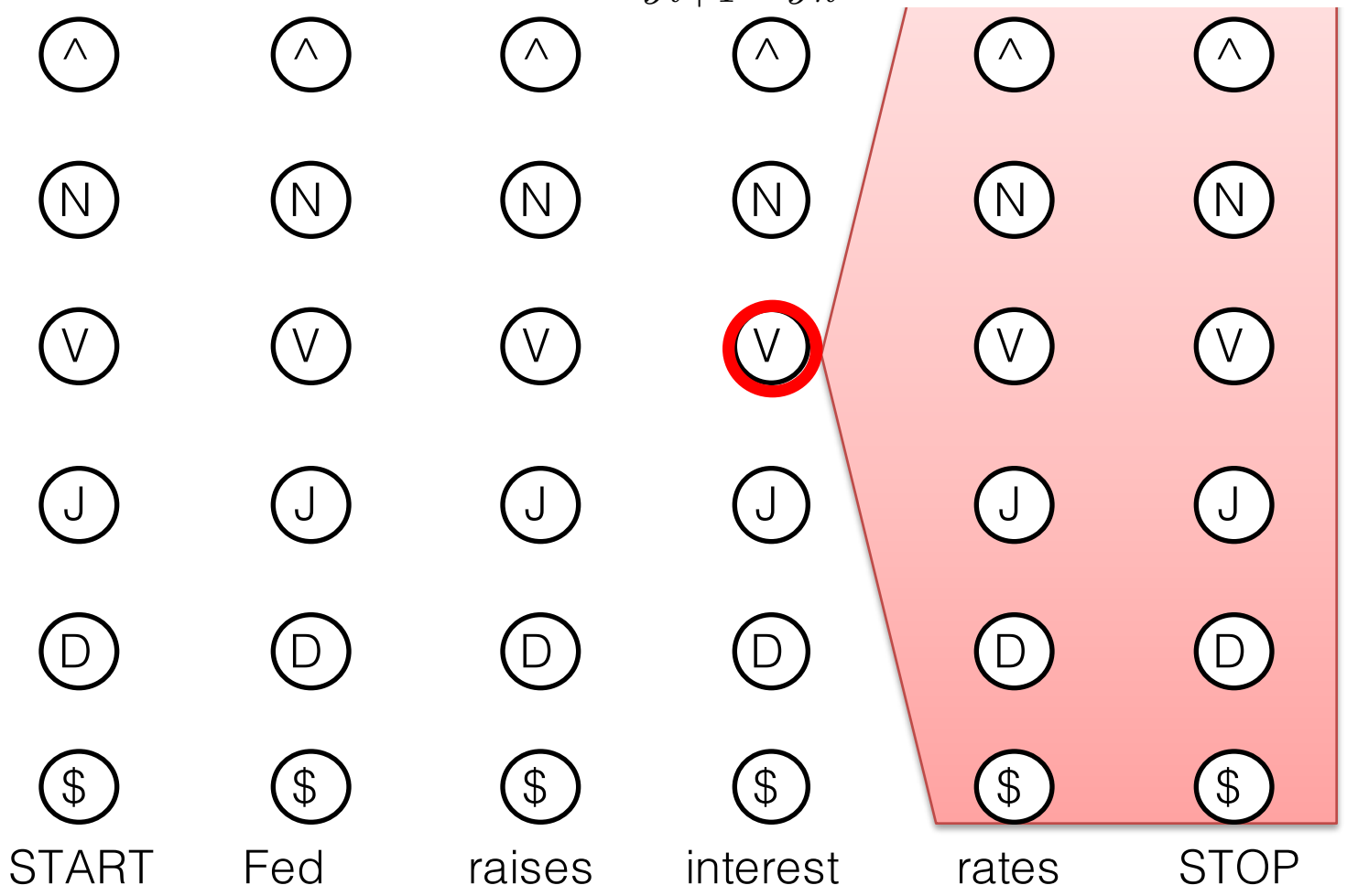
The State Lattice / Trellis: Forward

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



The State Lattice / Trellis: Backward

$$\beta(i, y_i) = p(x_{i+1} \dots x_n | y_i) = \sum_{y_{i+1} \dots y_n} p(x_{i+1} \dots x_n, y_{i+1} \dots y_n)$$



Forward Backward Algorithm

Two passes: one forward, one back:

Forward:

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

For i = 1 … 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:

$$\beta(n, y_n) = \begin{cases} 1 \text{ if } y_n == STOP \\ 0 \text{ otherwise} \end{cases}$$

For i = n-1 … 1:

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

Forward Backward: Runtime

- In term of sentence length n:
  - Linear
- In term of the number of states |K|:
  - Polynomial

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

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

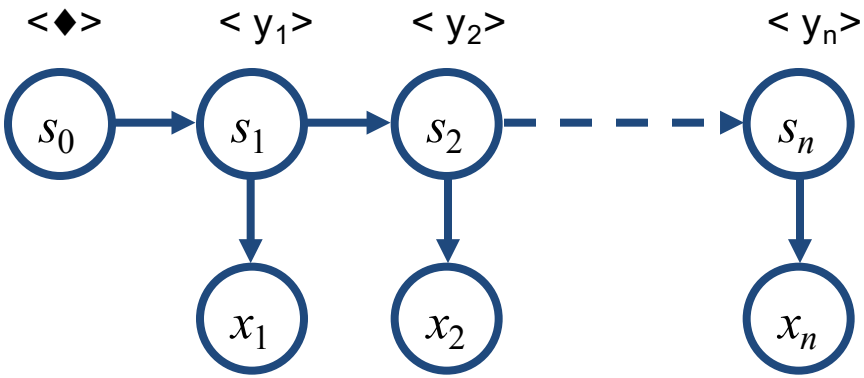
- Specifically:

$$O(n|\mathcal{K}|) \text{ entries in } \alpha(i, y_i) \text{ and } \beta(i, y_i)$$
$$O(|\mathcal{K}|) \text{ time to compute each entry}$$

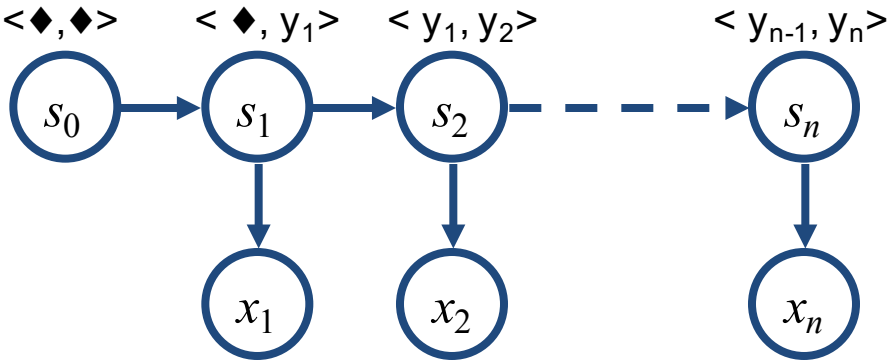
- Total runtime: $O(n|\mathcal{K}|^2)$
- Q: How does this compare to Viterbi?
- A: Asymptotically the same (actually x2, a constant factor)

What about n-gram Taggers?

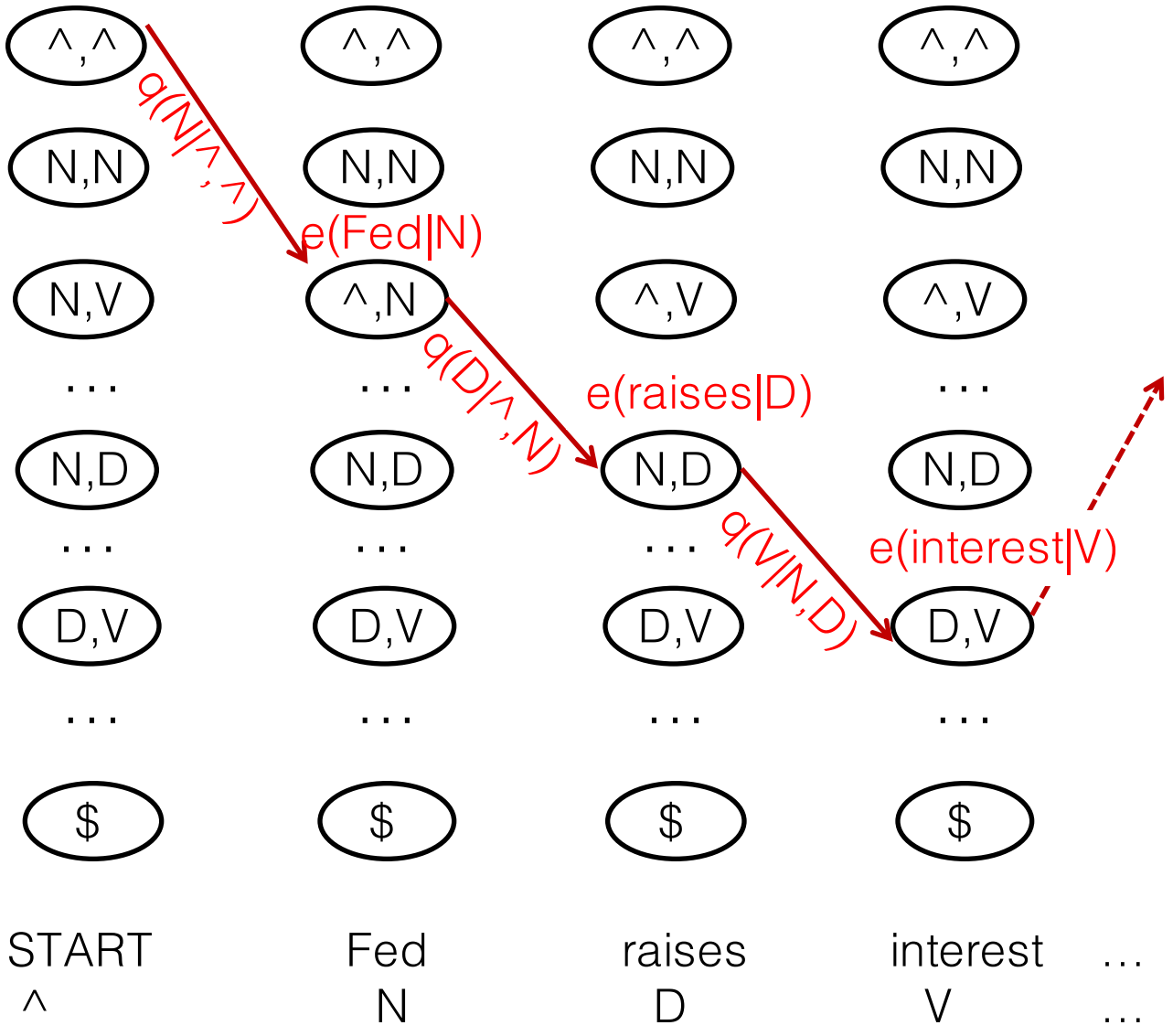
- States encode what is relevant about the past
- Transitions P(s | s') encode well-formed tag sequences
  - In a bigram tagger, states = tags



  - In a trigram tagger, states = tag pairs



The State Lattice / Trellis



Tagsets in Different Languages

Language	Source	# Tags
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	[illegible]
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	[illegible]
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	[illegible]
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	[illegible]
German	Tiger/CoNLL06 (Brants et al., 2002)	54
German	Negra (Skut et al., 1997)	54
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38
Hungarian	Szeged/CoNLL07 (Csendes et al., 2005)	43
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42
Korean	Sejong (http://www.sejong.or.kr)	187
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	[illegible]
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	[illegible]
Spanish	Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)	47
Swedish	Talbanken05/CoNLL06 (Nivre et al., 2006)	41
Turkish	METU-Sabanci/CoNLL07 (Oflazer et al., 2003)	31

$294^2 = 86436$

$294^4 = 7471182096$

$45^2 = 2045$

$45^4 = 4100625$

$11^2 = 121$

$11^4 = 14641$

[Petrov et al. 2012]

Some Numbers

Most errors on unknown words

- Rough accuracies:
  - Most freq tag:
  - Trigram HMM:
  - TnT (Brants, 2000):
    - A carefully smoothed trigram tagger
    - Suffix trees for emissions
    - 96.7% on WSJ text (SOA is ~97.5%)
    - Seen words accuracy: 97%
    - Percentage unknown: 2.9%