CS5740: Natural Language Processing
Spring 2018

Sequence Prediction and Part-of-speech Tagging

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Dan Jurafsky, Chris Manning, Michael Collins, Luke Zettlemoyer, Yejin Choi, and Slav Petrov

Overview

- POS Tagging: the problem
- Hidden Markov Models (HMM)
  - Supervised Learning
  - Inference
    - The Viterbi algorithm
- Feature-rich models
  - Maximum-entropy Markov Models
  - Perceptron
  - Conditional Random Fields

Parts of Speech

Open class (lexical) words

- Nouns
  - Proper: *IBM*, *Italy*
  - Common: *cat / cats*, *snow*
- Verbs
  - Main: *see*, *registered*
  - Modals: *can*, *had*
- Adjectives: *old older oldest*
- Adverbs: *slowly*
- Numbers: *122,312*, *one*
- *… more*

Closed class (functional)

- Determiners: *the some*
- Conjunctions: *and or*
- Pronouns: *he its*
- Prepositions: *to with*
- Particles: *off up*
- Interjections: *Ow Eh*
- *… more*

POS Tagging

- Words often have more than one POS: ***back***
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS Tagging

Penn Treebank POS tags

• Input:	Plays	well	with	others
• Ambiguity:	NNS/VBZ	UH/JJ/NN/RB	IN	NNS
• Output:	Plays/VBZ	well/RB	with/IN	others/NNS

- Uses:
  - Text-to-speech (how do we pronounce "lead"?)
  - Can write regular expressions like (Det) Adj* N+ over the output for phrases, etc.
  - As input to or to speed up a full parser
  - If you know the tag, you can back off to it in other tasks

Penn TreeBank Tagset

- Possible tags: 45
- Tagging guidelines: 36 pages
- Newswire text

Main Tags

CC	conjunction, coordinating	and both but either or
CD	numeral, cardinal	mid-1890 nine-thirty 0.5 one
DT	determiner	a all an every no that the
EX	existential there	there
FW	foreign word	gemeinschaft hund ich jeux
IN	preposition or conjunction, subordinating	among whether out on by if
JJ	adjective or numeral, ordinal	third ill-mannered regrettable
JJR	adjective, comparative	braver cheaper taller
JJS	adjective, superlative	bravest cheapest tallest
MD	modal auxiliary	can may might will would
NN	noun, common, singular or mass	cabbage thermostat investment subhumanity
NNP	noun, proper, singular	Motown Cougar Yvette Liverpool
NNPS	noun, proper, plural	Americans Materials States
NNS	noun, common, plural	undergraduates bric-a-brac averages
POS	genitive marker	' 's
PRP	pronoun, personal	hers himself it we them
PRP$	pronoun, possessive	her his mine my our ours their thy your
RB	adverb	occasionally maddeningly adventurously
RBR	adverb, comparative	further gloomier heavier less-perfectly
RBS	adverb, superlative	best biggest nearest worst
RP	particle	aboard away back by on open through
TO	"to" as preposition or infinitive marker	to
UH	interjection	huh howdy uh whammo shucks heck
VB	verb, base form	ask bring fire see take
VBD	verb, past tense	pleaded swiped registered saw
VBG	verb, present participle or gerund	stirring focusing approaching erasing
VBN	verb, past participle	dilapidated imitated reunifed unsettled
VBP	verb, present tense, not 3rd person singular	twist appear comprise mold postpone
VBZ	verb, present tense, 3rd person singular	bases reconstructs marks uses
WDT	WH-determiner	that what whatever which whichever
WP	WH-pronoun	that what whatever which who whom
WP$	WH-pronoun, possessive	whose
WRB	Wh-adverb	however whenever where why

Penn TreeBank Tagset

- How accurate are taggers? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of simplest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (*the*, *a*, etc.) and for punctuation marks!
  - Upperbound: probably 2% annotation errors

Hard Cases are Hard

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

How Difficult is POS Tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can't go *that* far = RB
- 40% of the word tokens are ambiguous

The Tagset

- Wait, do we really need all these tags?
- What about other languages?
  - Each language has its own tagset

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)	34
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	30
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)	22
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29
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

[Petrov et al. 2012]

The Tagset

- Wait, do we really need all these tags?
- What about other languages?
  - Each language has its own tagset
    - But why is this bad?
    - Differences in downstream tasks
    - Harder to do language transfer

Alternative: The Universal Tagset

- 12 tags:
  - NOUN, VERB, ADJ, ADV, PRON, DET, ADP, NUM, CONJ, PRT, '.', and X.
- Deterministic conversion from tagsets in 22 languages.
- Better unsupervised parsing results
- Was used to transfer parsers

[Petrov et al. 2012]

Sources of Information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP VB(D) DT NN NN
    - VB NN IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb….
- The latter proves the most useful, but the former also helps

Word-level Features

- Can do surprisingly well just looking at a word by itself:
  - Word — the: the $\rightarrow$ DT
  - Lowercased words: importantly $\rightarrow$ RB
  - Prefixes — unfathomable: un- $\rightarrow$ JJ
  - Suffixes — Importantly: -ly $\rightarrow$ RB
  - Capitalization — Meridian: CAP $\rightarrow$ NNP
  - Word shapes — 35-year: d-x $\rightarrow$ JJ

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:

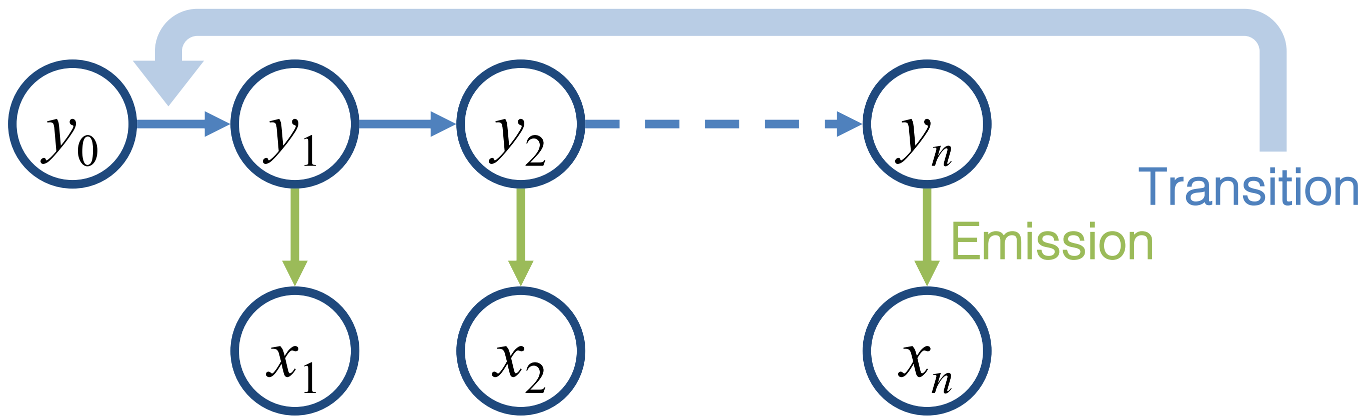
$$p(x_1 \ldots x_n, y_1 \ldots y_n)$$

And then compute the most likely assignment:

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

Classic Solution: HMMs

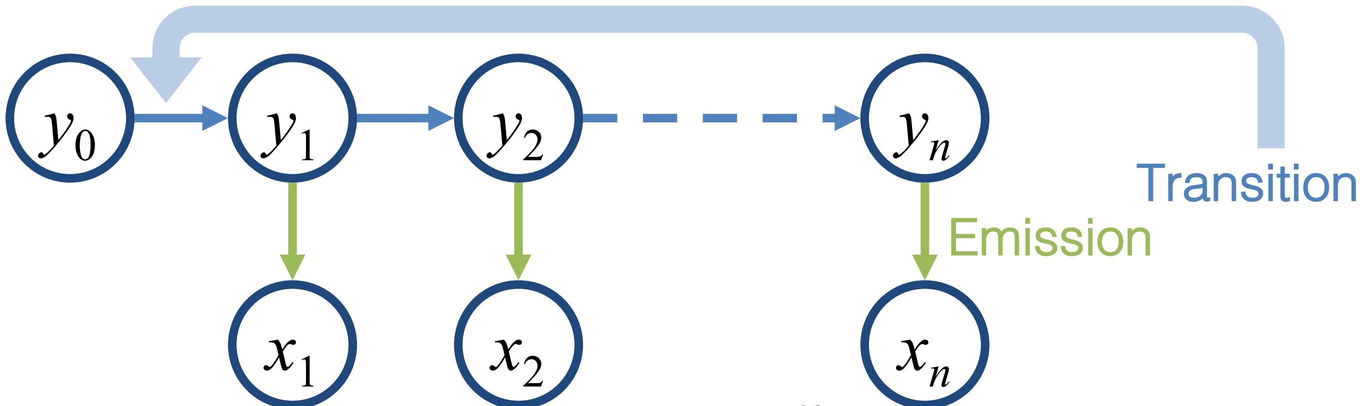
We want a model of sequences y and observations x



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

where y_0=*START* and we call $q(y_i \mid y_{i-1})$ the transition distribution and $e(x_i \mid y_i)$ the emission (or observation) distribution.

Model Assumptions



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

- Tag/state sequence is generated by a Markov model
- Words are chosen independently, conditioned only on the tag/state
- These are totally broken assumptions for POS: 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 $q(y_i|y_{i-1})$ models
  - Emission dist'n $e(x_i|y_i)$ models

HMM for POS Tagging

The Georgia branch had taken on loan commitments …

DT NNP NN VBD VBN RP NN NNS

- HMM Model:
  - States Y = {DT, NNP, NN, ... } are the POS tags
  - Observations X = V are words
  - Transition dist'n $q(y_i|y_{i-1})$ models the tag sequences
  - Emission dist'n $e(x_i|y_i)$ models words given their POS

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

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

  - Forward backward

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

Learning: Maximum Likelihood

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

- 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?

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

How is this a sequence tagging problem?

Named Entity Recognition

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…

- 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 $q(y_i|y_{i-1})$ models the tag sequences
  - Emission dist'n $e(x_i|y_i)$ 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/SO Co./CO ,/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

- NA = No entity
- SO = Start Organization
- CO = Continue Organization
- SL = Start Location
- CL = Continue Location
- …

Low Frequency Words: An Example

Profits/NA soared/NA at/NA Boeing/SO Co./CO ,/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
- SO = Start Organization
- CO = Continue Organization
- 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 . ➡ $logp(x,y) = -23$

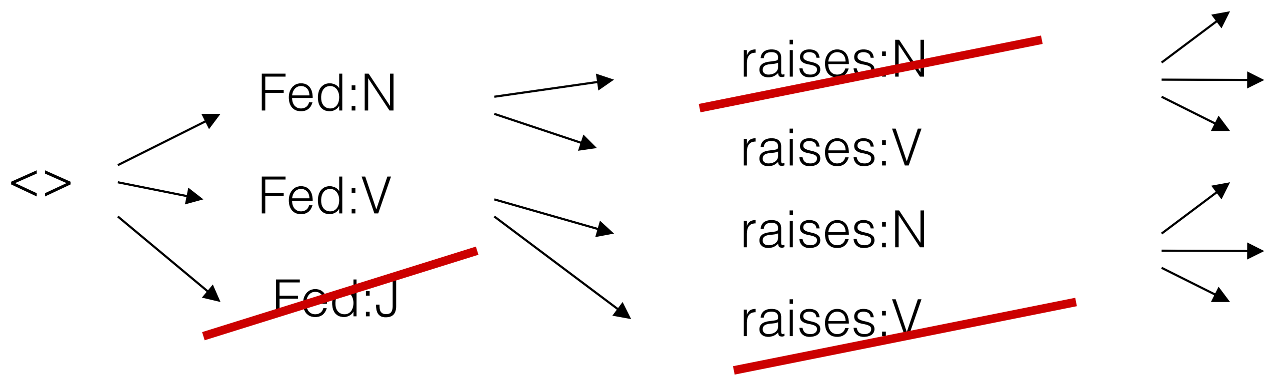
NNP NNS NN NNS CD NN . ➡ $\log(x,y) = -29$

NNP VBZ VB NNS CD NN . ➡ $\log p(x,y) = -27$

Any issue?

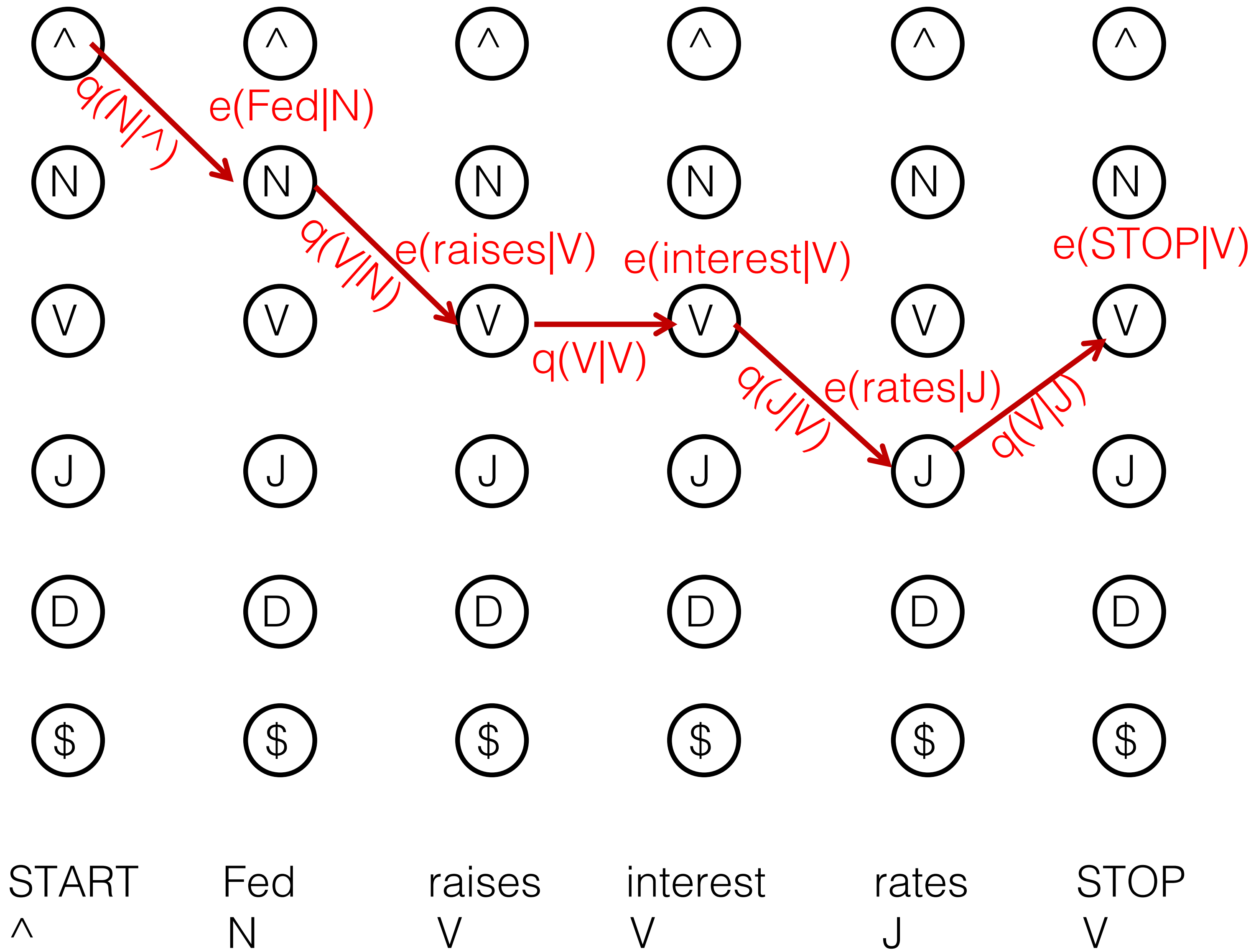
Finding the Best Trajectory

- 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 …
  - … sometimes you want the optimal answer
  - … and there’s usually a better option than naïve beams

The State Lattice / Trellis



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

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

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

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

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

The Viterbi Algorithm

Dynamic program for computing (for all i)

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

Iterative computation:

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

For i = 1 … n:

// Store score

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

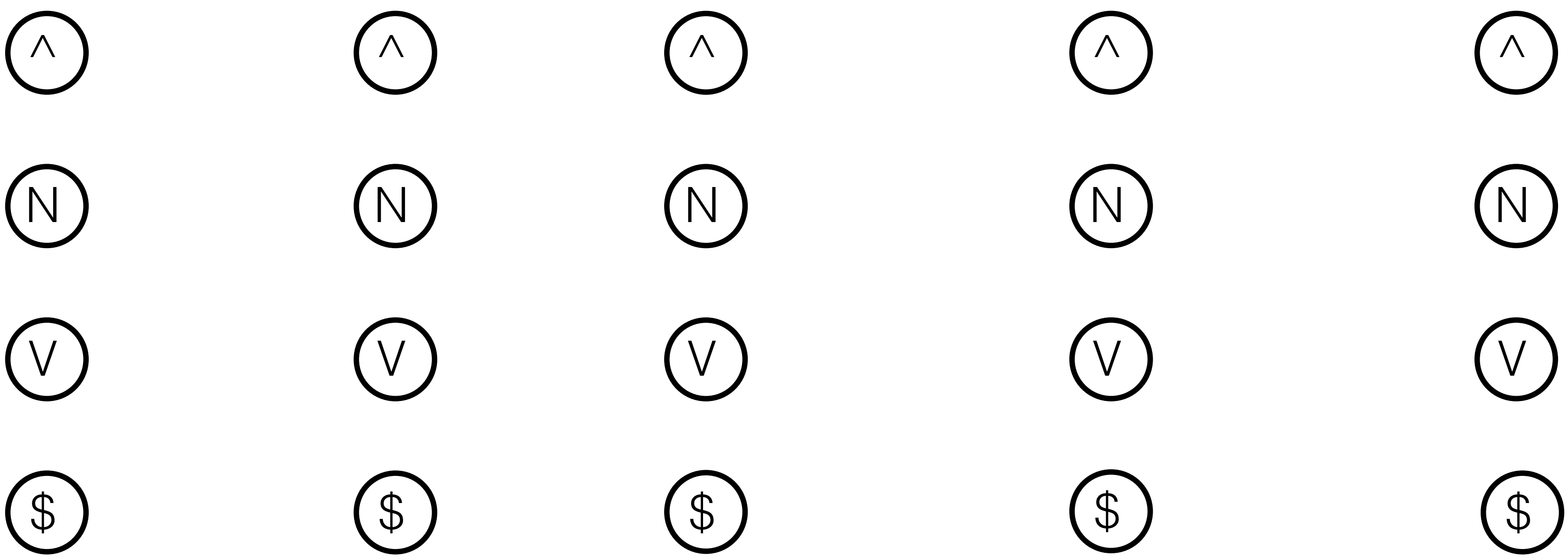
// Store back-pointer

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

What for?

The State Lattice / Trellis

Tie breaking:
Prefer first



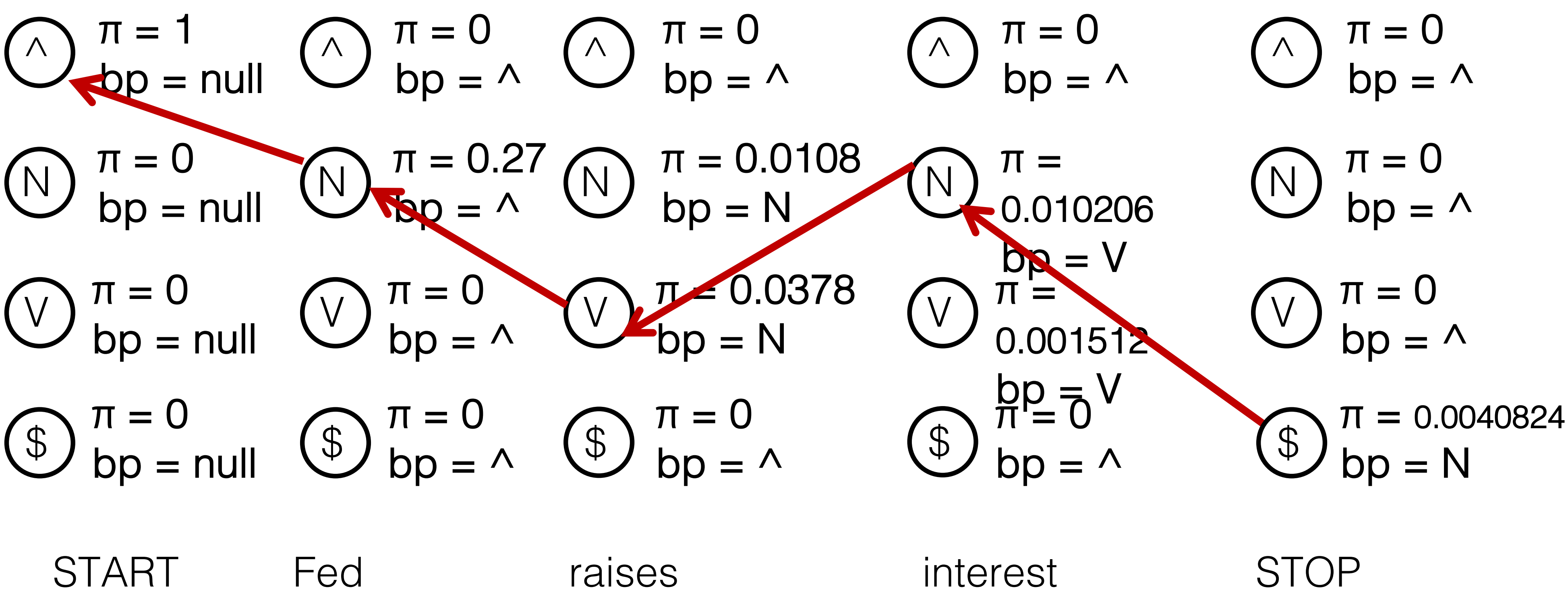
START Fed raises interest STOP

from \ to	^	N	V	$
^	0.0	0.6	0.4	0.0
N	0.0	0.4	0.2	0.4
V	0.0	0.6	0.1	0.3
$	0.0	0.0	0.0	1.0

emissions	START	Fed	raises	interest	STOP
^	1.0	0.0	0.0	0.0	0.0
N	0.0	0.45	0.1	0.45	0.0
V	0.0	0.0	0.7	0.4	0.0
$	0.0	0.0	0.0	0.0	1.0

Tie breaking:
Prefer first

The State Lattice / Trellis



from \ to	^	N	V	$
^	0.0	0.6	0.4	0.0
N	0.0	0.4	0.2	0.4
V	0.0	0.6	0.1	0.3
$	0.0	0.0	0.0	1.0

emissions	START	Fed	raises	interest	STOP
^	1.0	0.0	0.0	0.0	0.0
N	0.0	0.45	0.1	0.45	0.0
V	0.0	0.0	0.7	0.4	0.0
$	0.0	0.0	0.0	0.0	1.0

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)	34
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	30
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)	22
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29
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

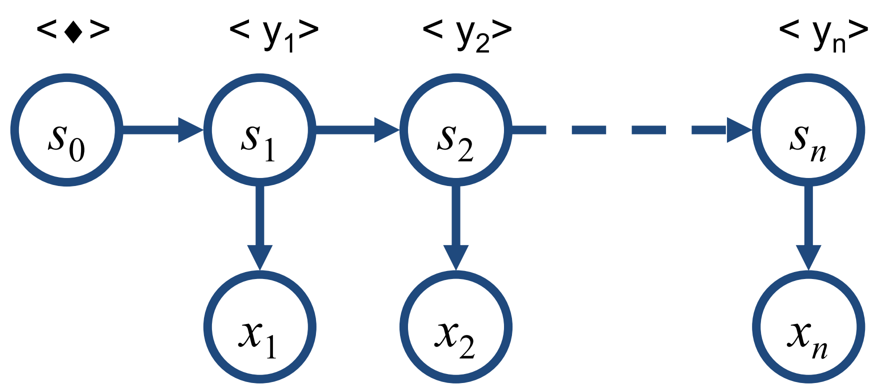
$$y^* = \arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

  - Forward backward

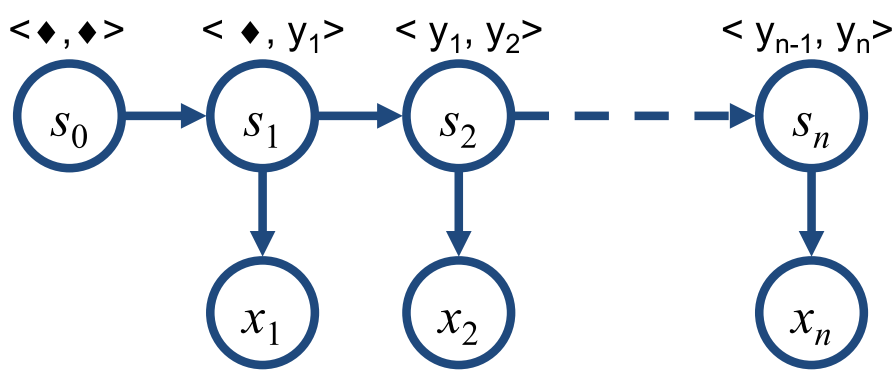
$$p(x_1 \dots x_n, y_i) = \sum_{y_1 \dots y_{i-1}} \sum_{y_{i+1} \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

What about n-gram Taggers?

- States encode what is relevant about the past
- Transitions $P(s_i \mid s_{i-1})$ encode well-formed tag sequences
  - In a bigram tagger, states = tags

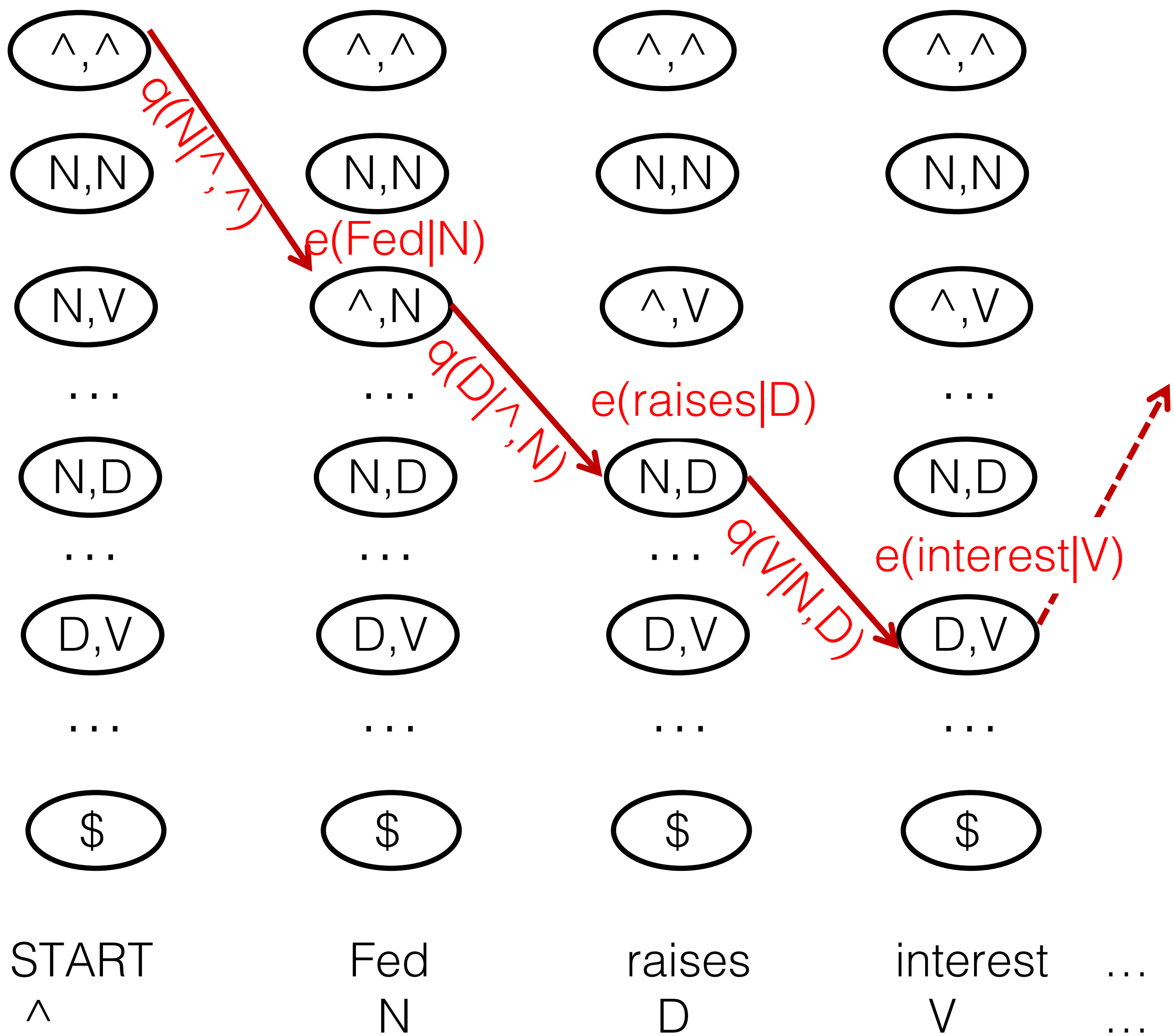


  - In a trigram tagger, states = tag pairs



The State Lattice / Trellis

Not all edges are allowed



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)	34
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	30
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)	22
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29
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
  - Upper bound: ~98%

Re-visit $P(x \mid y)$

- Reality check:
  - What if we drop the sequence?
    - Use only $P(x \mid y)$
  - Most frequent tag:
    - 90.3% with a so-so unknown word model
  - Can we do better?

What about better features?

- Looking at a word and its environment
  - Add in previous / next word   the __
  - Previous / next word shapes   X __ X
  - Occurrence pattern features   [X: x X occurs]
  - Crude entity detection   __ ….. (Inc.|Co.)
  - Phrasal verb in sentence?   put …… __
  - Conjunctions of these things

- Uses lots of features: > 200K

Some Numbers

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x)

- What does this tell us about sequence models?
- How do we add more features to our sequence models?

  - Upper bound: ~98%

MEMM Taggers

One step up: also condition on previous tags:

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

$$= \prod_{i=1}^{n} p(y_i | y_{i-1}, x_1 \dots x_n)$$

- Training:
  - Train $p(y_i | y_{i-1}, x_1 \dots x_n)$ as a discrete log-linear (MaxEnt) model
- Scoring:

$$p(y_i | y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$

- This is referred to as an MEMM tagger [Ratnaparkhi 96]

HMM vs. MEMM

- HMM models joint distribution:

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

- MEMM models conditioned distribution:

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

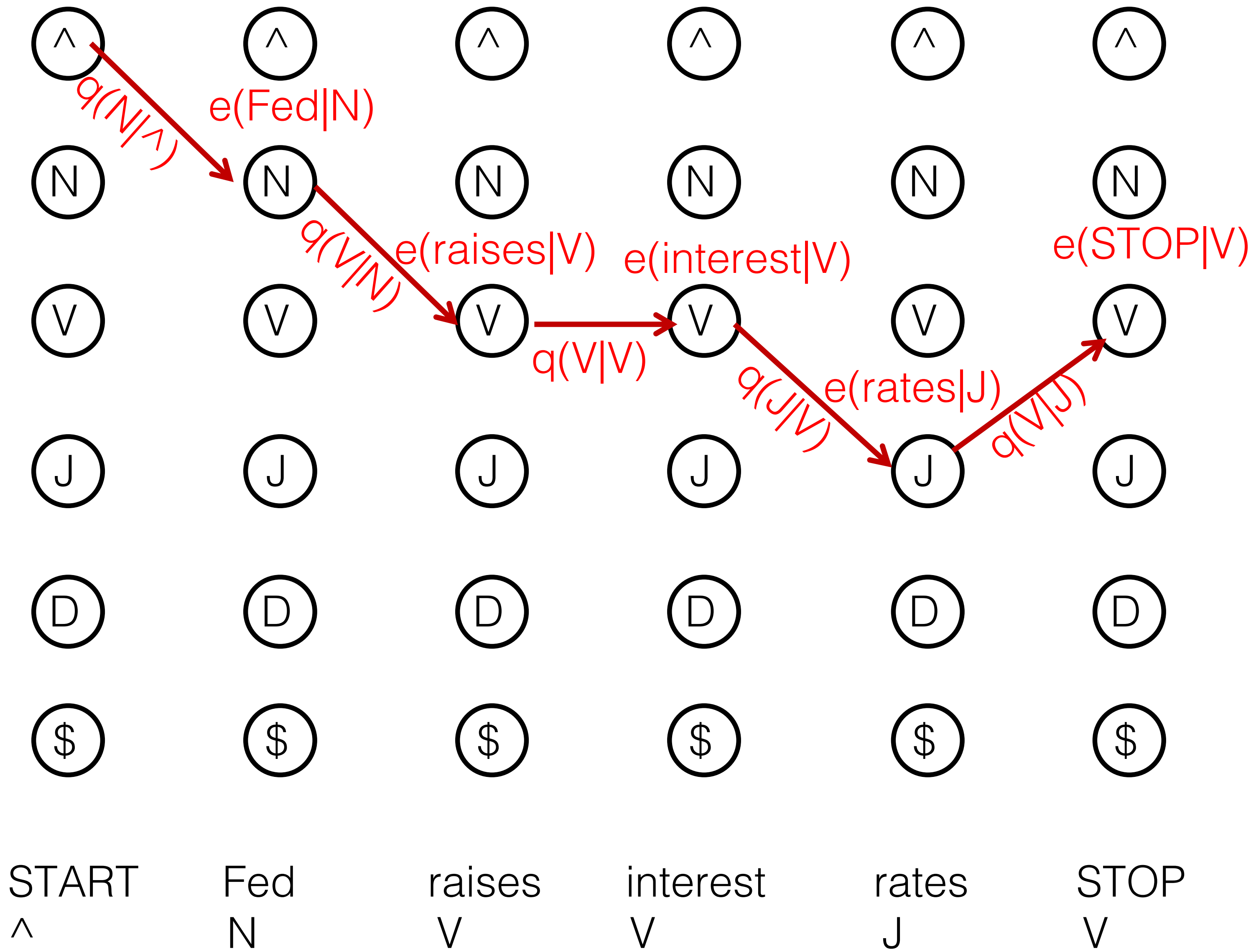
Decoding MEMM Taggers

- Scoring:

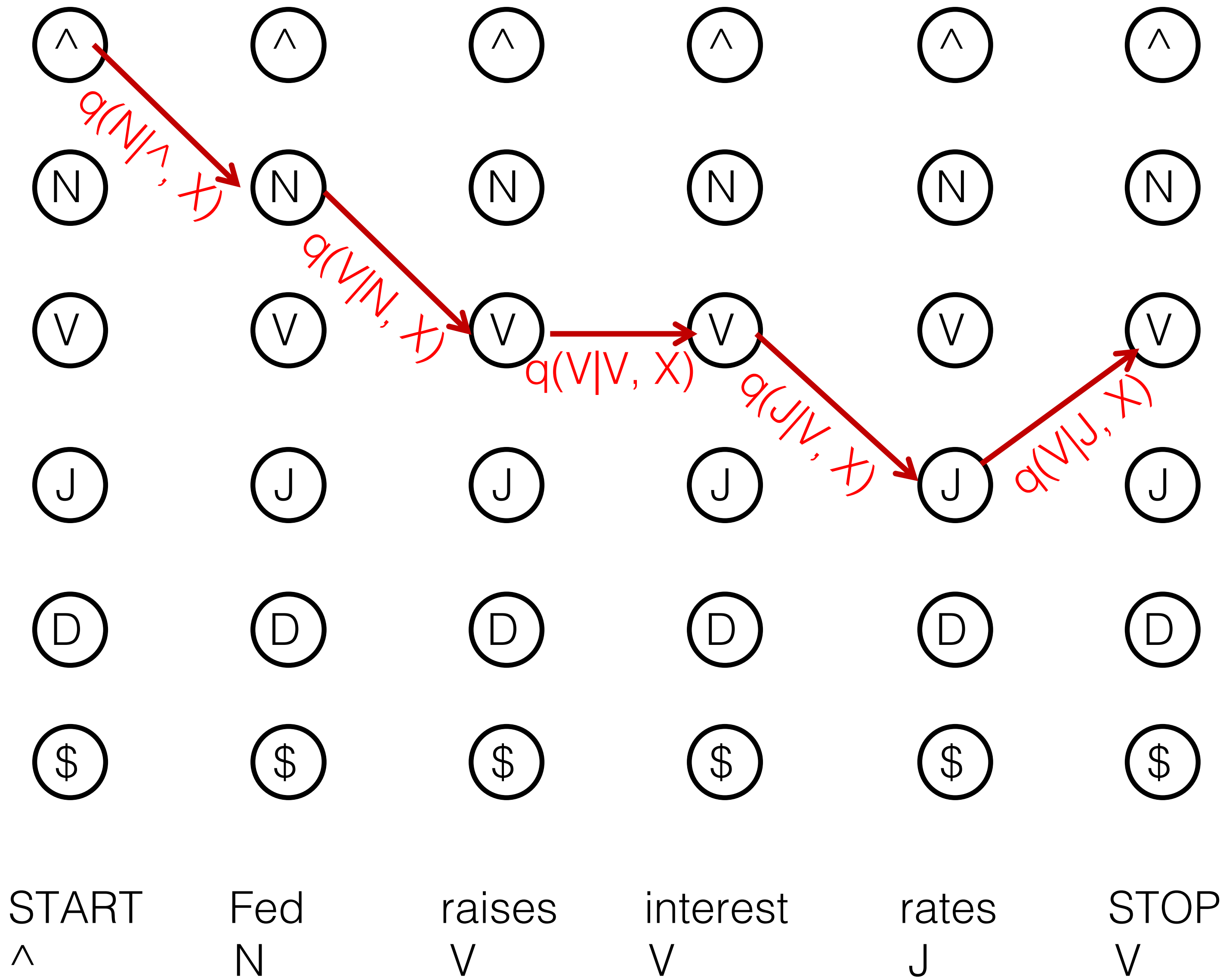
$$p(y_i|y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$

- Beam search is effective – why?
- Guarantees? Optimal?
- Can we do better?

The State Lattice / Trellis



The MEMM State Lattice / Trellis



Decoding MEMM Taggers

- Decoding MaxEnt taggers:
  - Just like decoding HMMs
  - Viterbi, beam search
- Viterbi algorithm (HMMs):
  - Define $\pi(i, yi)$ to be the max score of a sequence of length i ending in tag y_i

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

- Viterbi algorithm (MaxEnt):
  - Can use same algorithm for MEMMs, just need to redefine $\pi(i, yi)$!

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

Some Numbers

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x) 93.7% / 82.6%
  - MEMM tagger 1:

  - Upper bound: ~98%

[Ratnaparkhi 1996]

Feature Development

Common errors:

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	**177**	**56**	0	**61**	2	5	10	15	**108**	0	488
NN	**244**	0	**103**	0	12	1	1	29	5	6	19	525
NNP	**107**	**106**	0	**132**	5	0	7	5	1	2	0	427
NNPS	1	0	**110**	0	0	0	0	0	0	0	0	142
RB	**72**	21	7	0	0	16	**138**	1	0	0	0	295
RP	0	0	0	0	**39**	0	**65**	0	0	0	0	104
IN	11	0	1	0	**169**	**103**	0	1	0	0	0	323
VB	17	**64**	9	0	2	0	1	0	4	7	**85**	189
VBD	10	5	3	0	0	0	0	3	0	**143**	2	166
VBN	**101**	3	3	0	0	0	0	3	**108**	0	1	221
VBP	5	34	3	1	1	0	2	**49**	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

NN/JJ NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

[Toutanova and Manning 2000]

Some Numbers

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x) 93.7% / 82.6%
  - MEMM tagger 1: 96.7% / 84.5%
  - MEMM tagger 2:

  - Upper bound: ~98%

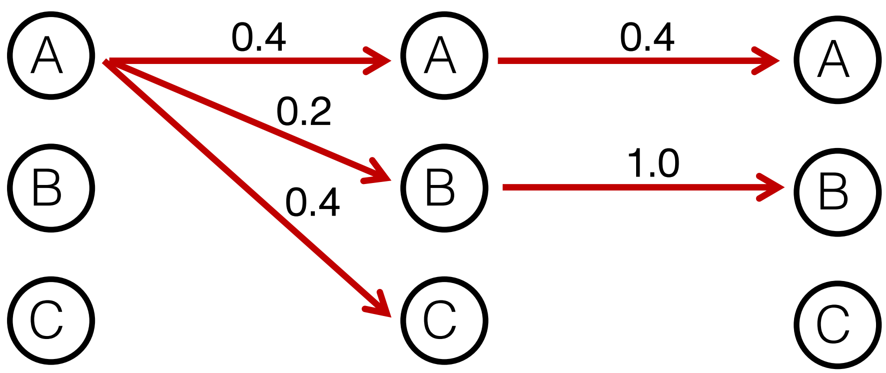
[Toutanova and Manning 2000]

Locally Normalized Models

- So far:
  - Probabilities are product of **locally normalized** probabilities
  - Is this bad?
- Label bias
  - States with fewer transitions are likely to be preferred because normalization is local
  - Extreme case: What happens if there is only one outgoing arc? Does it matter what the observation is?

Locally Normalized Models

- So far:
  - Probabilities are product of **locally normalized** probabilities
  - Is this bad?



AAA →
ABB →

from \ to	A	B	C
A	0.4	0.2	0.4
B	0.0	1.0	0.0
C	0.6	0.2	0.2

B → B transitions are likely to take over even if rarely observed!

Global Discriminative Taggers

- Discriminative sequence models
  - CRFs (also Perceptrons)
  - Do not decompose training into independent local regions
  - Can be very slow* to train – require repeated inference on training set

* Relatively slow. NN models are much slower.

Linear Models: Perceptron

- The perceptron algorithm
  - Iteratively processes the data, reacting to training errors
  - Can be thought of as trying to drive down training error
- The (online structured) perceptron algorithm:
  - Start with zero weights
  - Visit training instances $(X^{(i)}, Y^{(i)})$ one by one
    - Make a prediction

$$Y^* = \arg\max_Y w \cdot \phi(X^{(i)}, Y)$$

Sentence: $X = x_1 \dots x_n$

Tag Sequence: $Y = y_1 \dots y_m$

    - If correct ($Y^* == Y^{(i)}$):
      - no change, goto next example!
    - If wrong:
      - adjust weights: $w = w + \phi(X^{(i)}, Y^{(i)}) - \phi(X^{(i)}, y^*)$

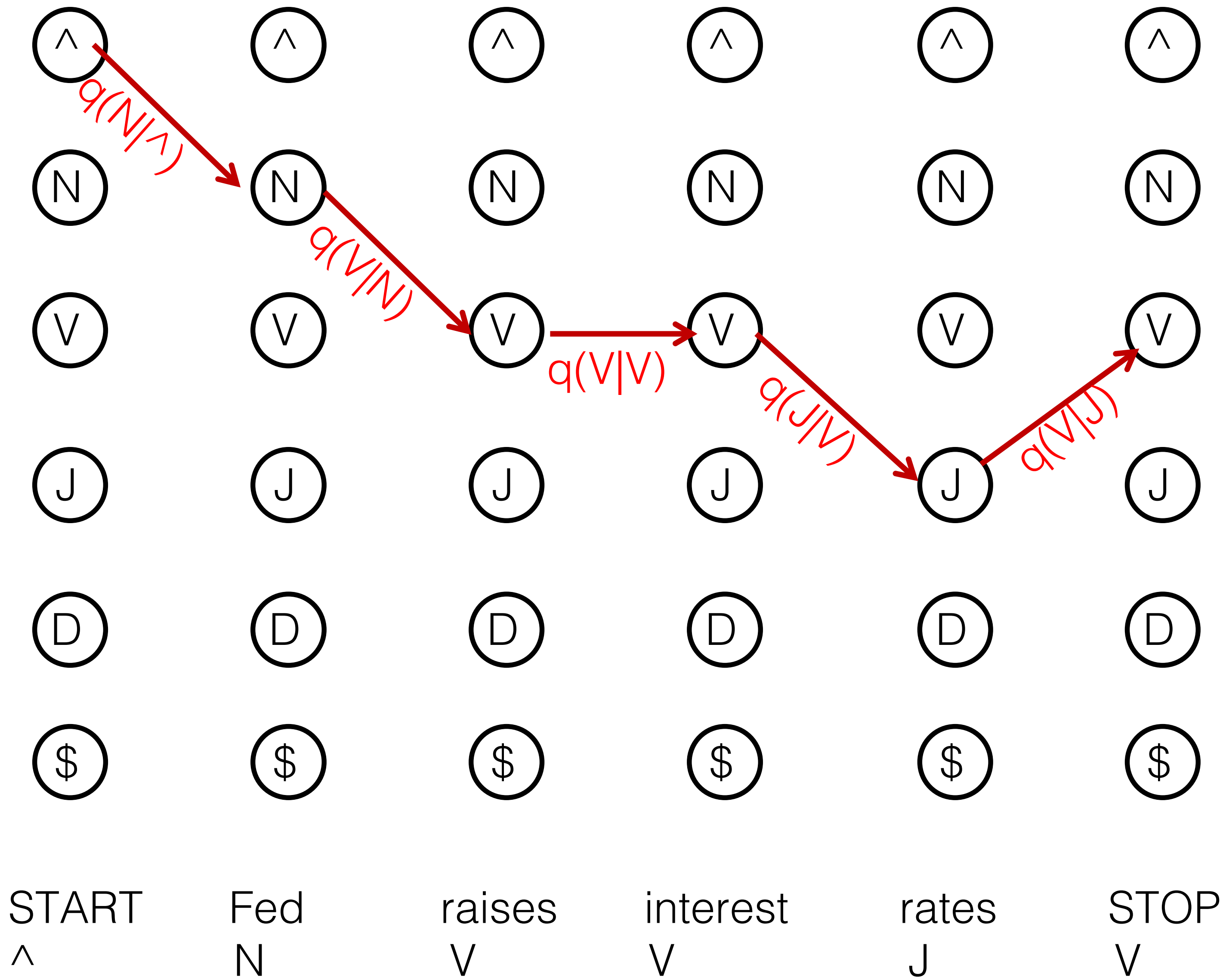
- Challenge: How to compute argmax efficiently?

Decoding

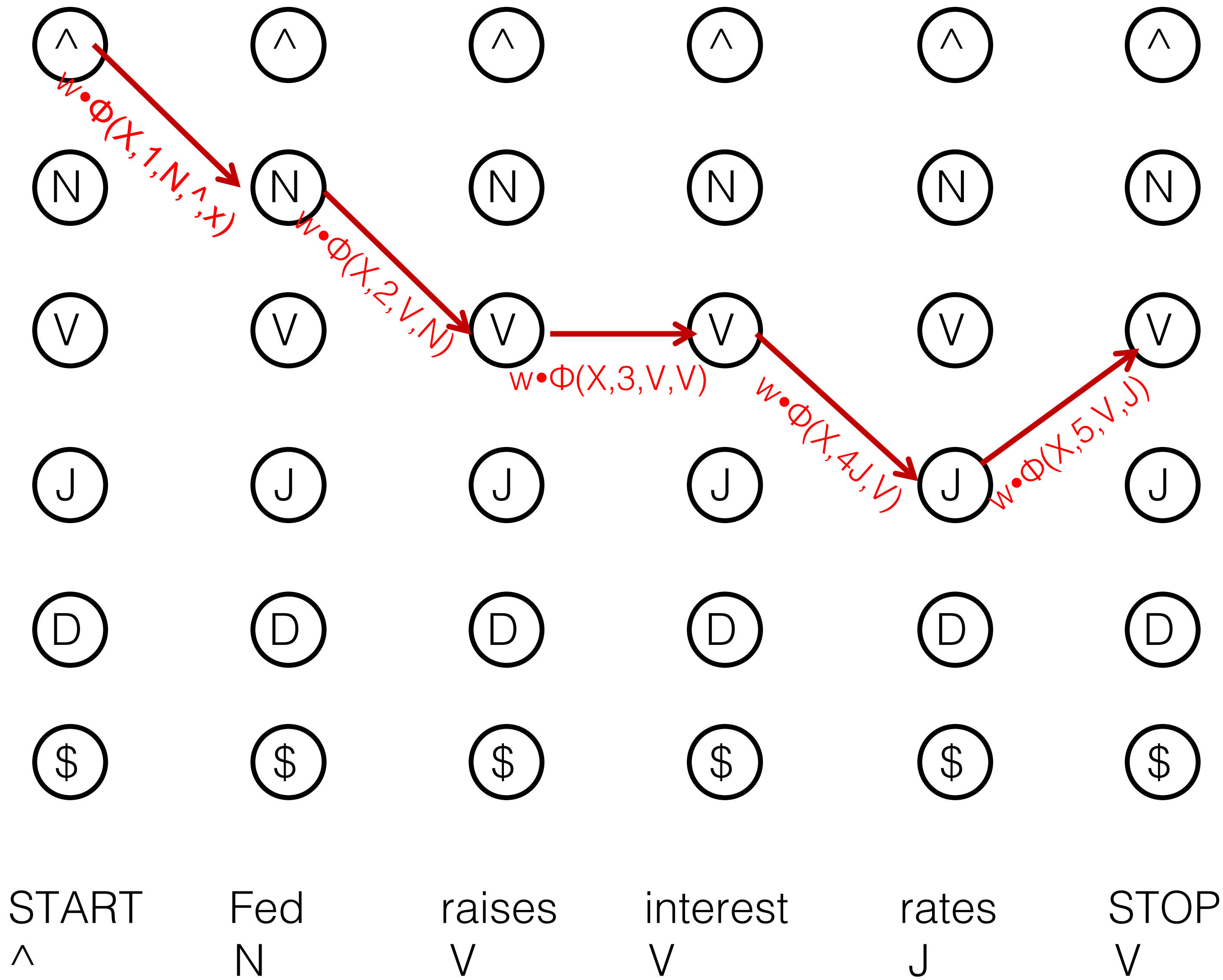
- Linear Perceptron $Y^* = \arg\max_Y w \cdot \phi(X, Y)$
  - Features must be local, for $X = x_1 \dots x_n$, and $Y = y_1 \dots y_m$

$$\phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

The MEMM State Lattice / Trellis



The Perceptron State Lattice / Trellis



Decoding

- Linear Perceptron $Y^* = \arg\max_Y w \cdot \phi(X, Y)$
  - Features must be local, for $X = x_1 \dots x_n$, and $Y = y_1 \dots y_n$

$$\phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

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

$$\pi(i, y_i) = \max_{y_{i-1}} w \cdot \phi(X, i, y_{i-1}, y_i) + \pi(i-1, y_{i-1})$$

- Viterbi algorithm (HMMs):

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

- Viterbi algorithm (Maxent):

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

Some Numbers

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x) 93.7% / 82.6%
  - MEMM tagger 1: 96.7% / 84.5%
  - MEMM tagger 2: 96.8% / 86.9%
  - Perceptron:

  - Upper bound: ~98%

[Collins 2002]

Conditional Random Fields (CRFs)

- What did we lose with the Perceptron?
  - No probabilities
  - Let’s try again with a probabilistic model

CRFs

- Maximum entropy (logistic regression)

Sentence: $X = x_1 \ldots x_n$

$$p(Y \mid X; w) = \underline{\hspace{8cm}}$$

Tag Sequence: $Y = y_1 \ldots y_n$

  - Learning: maximize the (log) conditional likelihood of training data $\{(X^{(i)}, Y^{(i)})\}_{i=1}^{m}$

$$\frac{\partial}{\partial w_j} L(w) = \sum_{I=1}^{m} \left(\phi_j(X^{(i)}, Y^{(i)}) - \sum_{Y} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, Y) \right) - \lambda w_j$$

  - Computational challenges?
    - Most likely tag sequence, normalization constant, gradient

[Lafferty et al. 2001]

Decoding

- CRFs

$$Y^* = \arg\max_Y p(Y \mid X; w)$$

  – Features must be local, for $x = x_1 \ldots x_n$, and $y = y_1 \ldots y_n$

$$p(Y \mid X; w) = \frac{\exp(w \cdot \phi(X,Y))}{\sum_{Y'} \exp(w \cdot \phi(X,Y'))} \quad \phi(X,Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

$$\arg\max_Y \frac{\exp(w \cdot \phi(X,Y))}{\sum_{Y'} \exp(w \cdot \phi(X,Y'))} = \arg\max_Y \exp(w \cdot \phi(X,Y))$$
$$= \arg\max_Y w \cdot \phi(X,Y)$$

- Looks familiar?
- Same as linear Perceptron!

$$\pi(i, y_i) = \max_{y_{i-1}} \phi(x, i, y_{i-i}, y_i) + \pi(i-1, y_{i-1})$$

CRFs: Computing Normalization

$$p(Y \mid X; w) = \frac{\exp(w \cdot \phi(X, Y))}{\sum_{Y'} \exp(w \cdot \phi(X, Y'))} \quad \phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

$$\sum_{Y'} \exp(w \cdot \phi(X, Y')) = \sum_{Y'} \exp\left(\sum_{j=1}^{n} w \cdot \phi(X, j, y_{j-1}, y_j)\right)$$

$$= \sum_{Y'} \prod_{j=1}^{n} \exp(w \cdot \phi(X, j, y_{j-1}, y_j))$$

Define $norm(i, y_i)$ to sum of scores for sequences ending in position i

$$\mathrm{norm}(i, y_i) = \sum_{y_{i-1}} \exp(w \cdot \phi(X, i, y_{i-1}, y_i))\mathrm{norm}(i-1, y_{i-1})$$

- Forward algorithm! Remember HMM case:

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

CRFs: Computing Gradient

$$p(Y \mid X; w) = \frac{\exp(w \cdot \phi(X, Y))}{\sum_{Y'} \exp(w \cdot \phi(X, Y'))} \quad \phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

$$\frac{\partial}{\partial w_j} L(w) = \sum_{I=1}^{m} \left(\phi_j(X^{(i)}, Y^{(i)}) - \sum_{Y} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, Y) \right) - \lambda w_j$$

$$\begin{aligned} \sum_{Y} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, Y) &= \sum_{Y} p(Y \mid X^{(i)}; w) \sum_{k=1}^{n} \phi_j(X^{(i)}, k, y_{k-1}, y_k) \\ &= \sum_{k=1}^{n} \sum_{a,b} \sum_{y_{k-1}=a, y_k=b} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, k, y_{k-1}, y_k) \end{aligned}$$

- Can compute with the Forward Backward algorithm

  See notes for full details!

Some Numbers

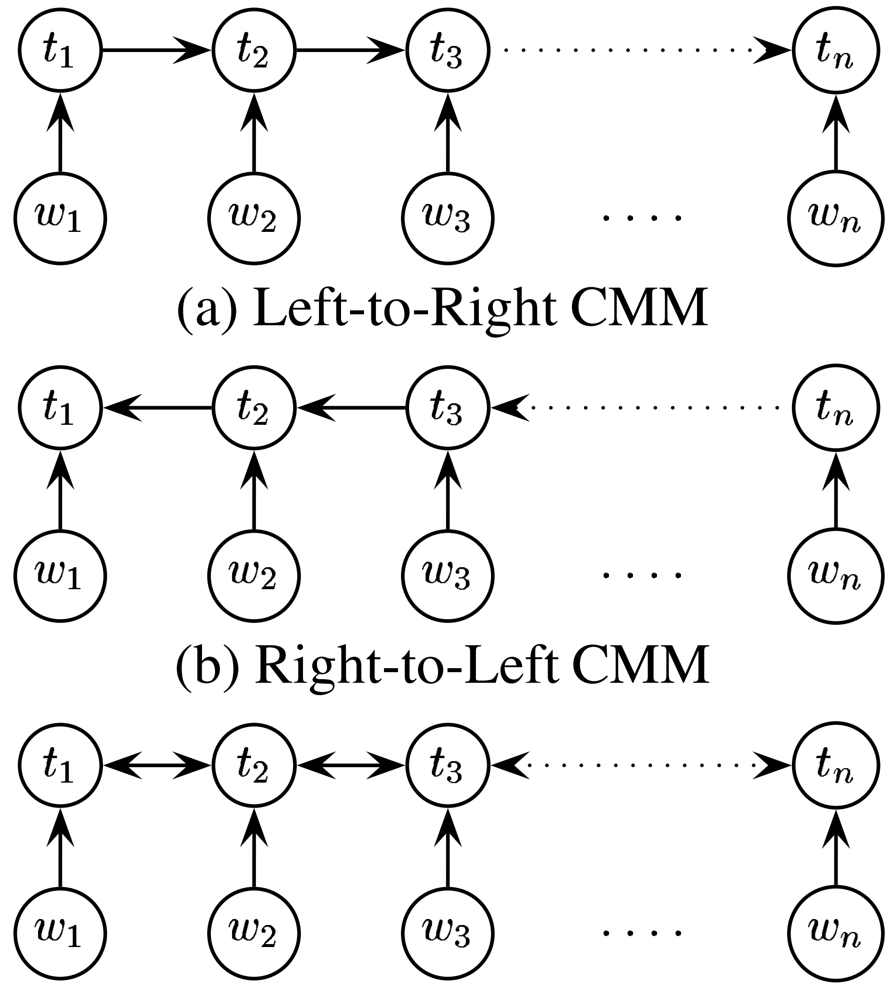
- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x) 93.7% / 82.6%
  - MEMM tagger 1: 96.7% / 84.5%
  - MEMM tagger 2: 96.8% / 86.9%
  - Perceptron: 97.1%
  - CRF++:

  - Upper bound: ~98%

[Sun 2014]

Cyclic Network

- Train two MEMMs, combine scores
- And be very careful
  - Tune regularization
  - Try lots of different features
  - See paper for full details



(a) Left-to-Right CMM

(b) Right-to-Left CMM

(c) Bidirectional Dependency Network

[Toutanova et al. 2003]

Some Numbers

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x) 93.7% / 82.6%
  - MEMM tagger 1: 96.7% / 84.5%
  - MEMM tagger 2: 96.8% / 86.9%
  - Perceptron: 97.1%
  - CRF++: 97.3%
  - Cyclic tagger:
  - Upper bound: ~98%

[Toutanova et al. 2003]

Summary

- For tagging, the change from **generative to discriminative** model does not by itself result in great improvement
- One profits from models for specifying dependence on **overlapping features** of the observation such as spelling, suffix analysis, etc.
- MEMMs allow **integration of rich features** of the observations
- This **additional power** (of the MEMM ,CRF, Perceptron models) has been shown to result in improvements in accuracy
- The higher accuracy of discriminative models comes at the price of **much slower training**

Domain Effects

- Accuracies degrade outside of domain
  - Up to triple error rate
  - Usually make the most errors on the things you care about in the domain (e.g. protein names)

- Open questions
  - How to effectively exploit unlabeled data from a new domain (what could we gain?)
  - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)