CSE 517
Natural Language Processing
Winter2015

Feature Rich Models

Sameer Singh

Guest lecture for Yejin Choi - University of Washington

[Slides from Jason Eisner, Dan Klein, Luke Zettlemoyer]

Outline

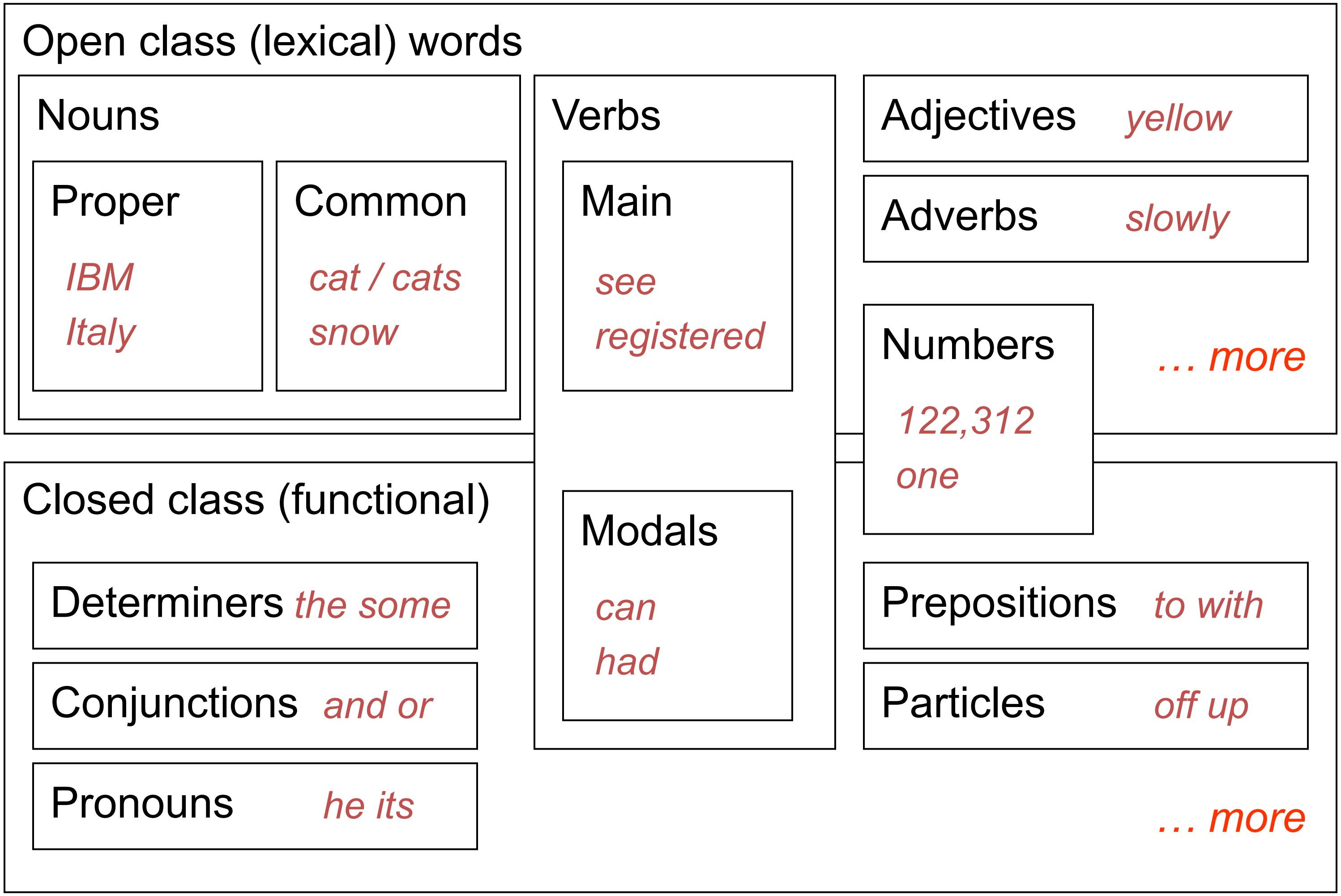
- POS Tagging
- MaxEnt
- MEMM
- CRFs
- Wrap-up
- Optional: Perceptron

Outline

- POS Tagging
- MaxEnt
- MEMM
- CRFs
- Wrap-up
- Optional: Perceptron

Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes



Open class (lexical) words

Nouns

Proper *IBM* *Italy*

Common *cat / cats* *snow*

Verbs

Main *see* *registered*

Modals *can* *had*

Adjectives *yellow*

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*

… more

Penn Treebank POS: 36 possible tags, 34 pages of tagging guidelines.

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" as preposition or infinitive	

ftp://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz

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

ftp://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz

Why POS Tagging?

- Useful in and of itself (more than you’d think)
  - Text-to-speech: record, lead
  - Lemmatization: saw[v] ⍰ see, saw[n] ⍰ saw v adj
  - Quick-and-dirty NP-chunk detection: grep {JJ | NN}* {NN | NNS}

- Useful as a pre-processing step for parsing
  - Less tag ambiguity means fewer parses
  - However, some tag choices are better decided by parsers

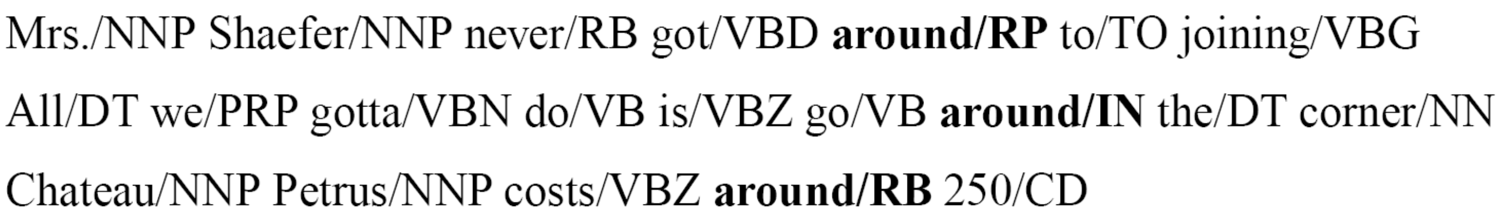
IN
DT NNP NN VBD VBN RP NN NNS
The Georgia branch had taken on loan commitments …

VDN
DT NN IN NN VBD NNS VBD
The average of interbank offered rates plummeted …

Part-of-Speech Ambiguity

- Words can have multiple parts of speech

Fed raises interest rates



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

Part-of-Speech Ambiguity

- Words can have multiple parts of speech

VBD		VB	
VBN	NNS	VBP	VBZ
NNP	VBZ	NN	NNS
Fed	raises	interest	rates

Ambiguity in POS Tagging

- **Particle (RP) vs. preposition (IN)**
  - He talked *over* the deal. — RP
  - He talked *over* the telephone. — IN
- **past tense (VBD) vs. past participle (VBN)**
  - The horse *walked* past the barn.
  - The horse *walked* past the barn fell.
- **noun vs. adjective?**
  - The *executive* decision.
- **noun vs. present participle**
  - *Fishing* can be fun

Ambiguity in POS Tagging

- "Like" can be a verb or a preposition
  - I **like/VBP** candy.
  - Time flies **like/IN** an arrow.

- "Around" can be a preposition, particle, or adverb
  - I bought it at the shop **around/IN** the corner.
  - I never got **around/RP** to getting a car.
  - A new Prius costs **around/RB** $25K.

adverb

Baselines and Upper Bounds

- Choose the most common tag
  - ~~90.3%~~ with a bad unknown word model
  - 93.7% with a good one

- Noise in the data
  - Many errors in the training and test corpora
  - Probably about 2% guaranteed error from noise (on this data)

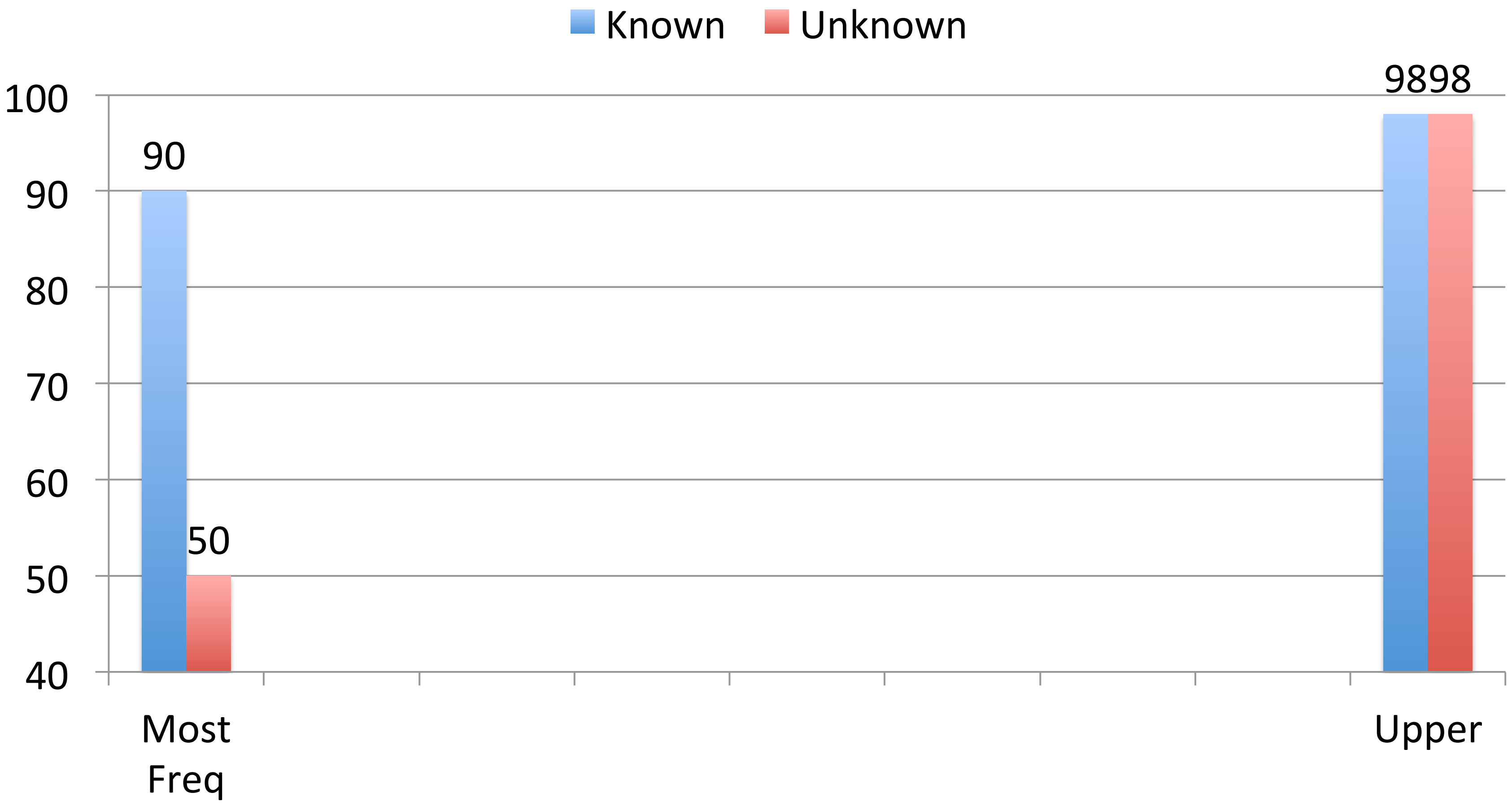
JJ JJ NN
chief executive officer

NN JJ NN
chief executive officer

JJ NN NN
chief executive officer

NN NN NN
chief executive officer

POS Results

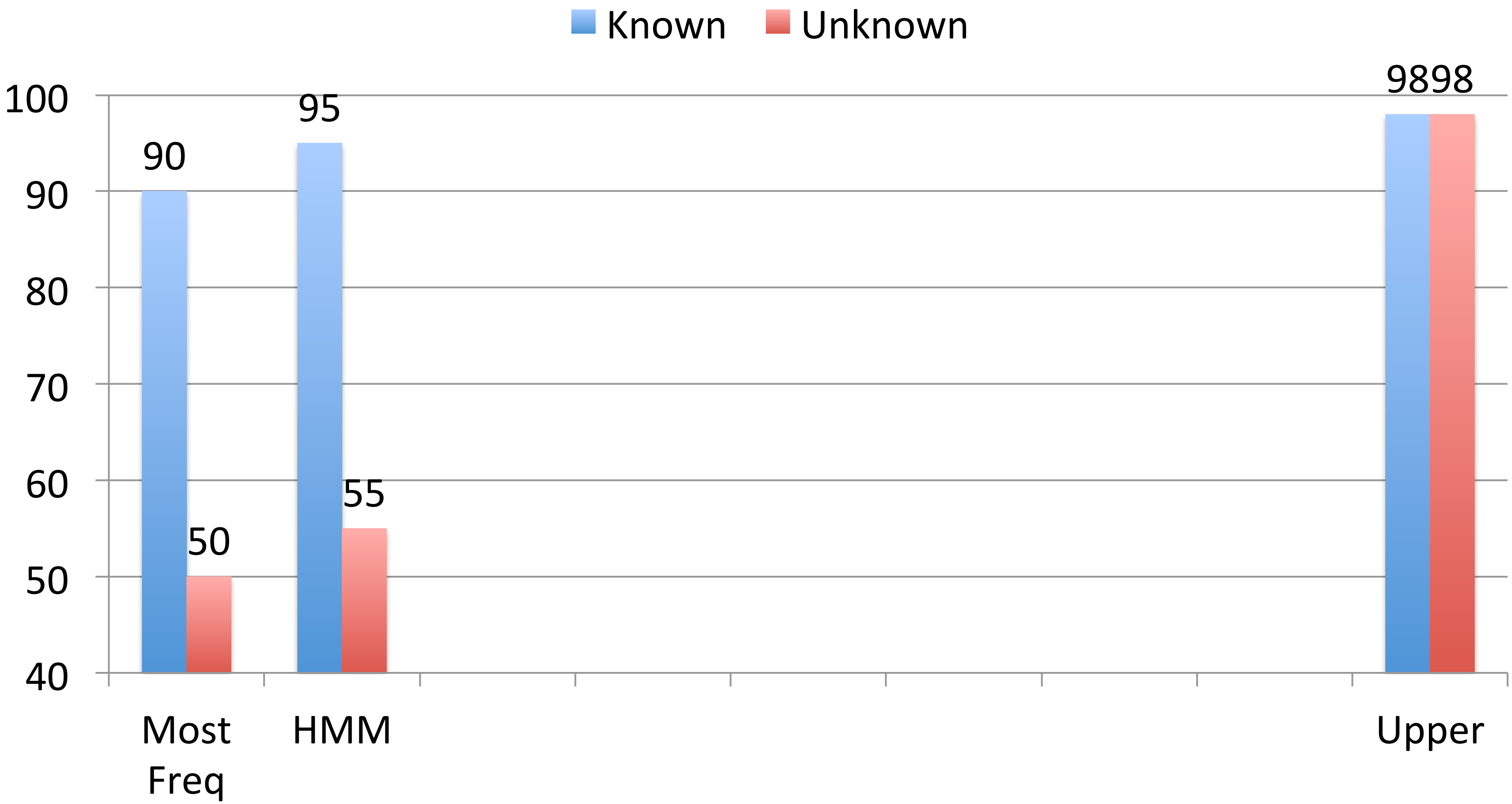


Overview: Accuracies

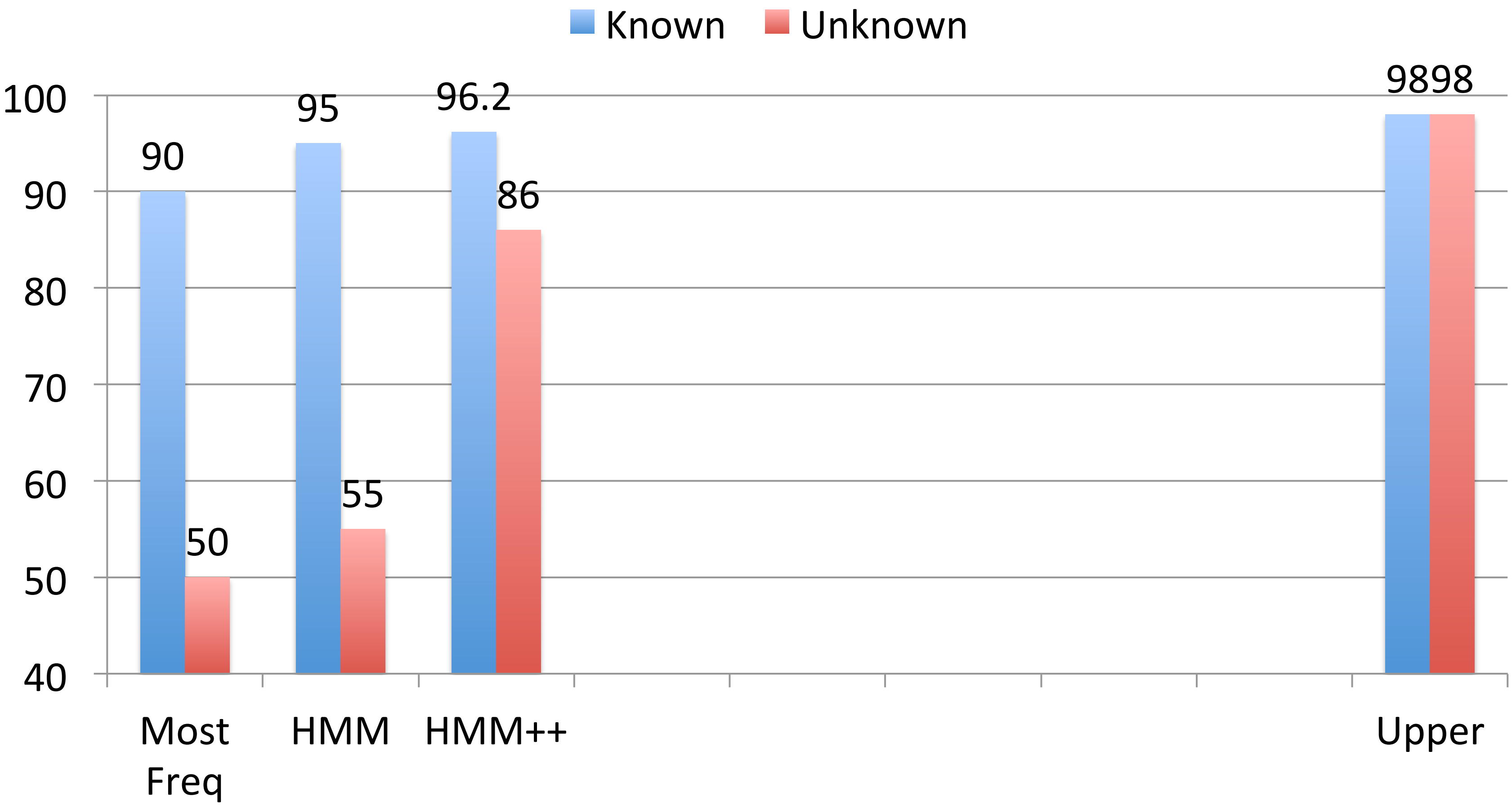
- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text (SOA is ~97.5%)
  - Upper bound: ~98%

Most errors on unknown words

POS Results


Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
9898
Most Freq
HMM
Upper

POS Results


Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
96.2
86
9898
Most Freq
HMM
HMM++
Upper

Outline

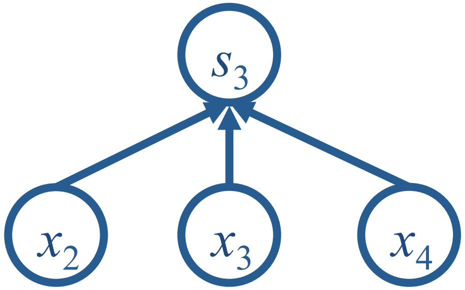
- POS Tagging
- MaxEnt
- MEMM
- CRFs
- Wrap-up
- Optional: Perceptron

What about better features?

- Choose the most common tag
  - 90.3% with a bad unknown word model
  - 93.7% with a good one

- What about looking at a word and its environment, but no sequence information?

– 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 ~ 20 million

3.

Maximum Entropy (MaxEnt) Models

- Also known as "Log-linear" Models (*linear if you take log*)

pos sentence/word weights features

$$P(\mathbf{y}_i|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y},x))}{\sum_{\mathbf{y}'}\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}',x))}$$

$f(x) =$ is Capitalized

$f(\text{Sameer}) = 1$

$f(\text{Sameer}, N)$ $\begin{array}{cc} N & V \\ \hline ▨ & 0 \end{array}$

$f(\text{Sameer}, V)$ $\begin{array}{cc} N & V \\ \hline 0 & ▨ \end{array}$ f(Sameer)

$f(x,y) =$ is Capitalized and y

- The feature vector representation may include redundant and overlapping features

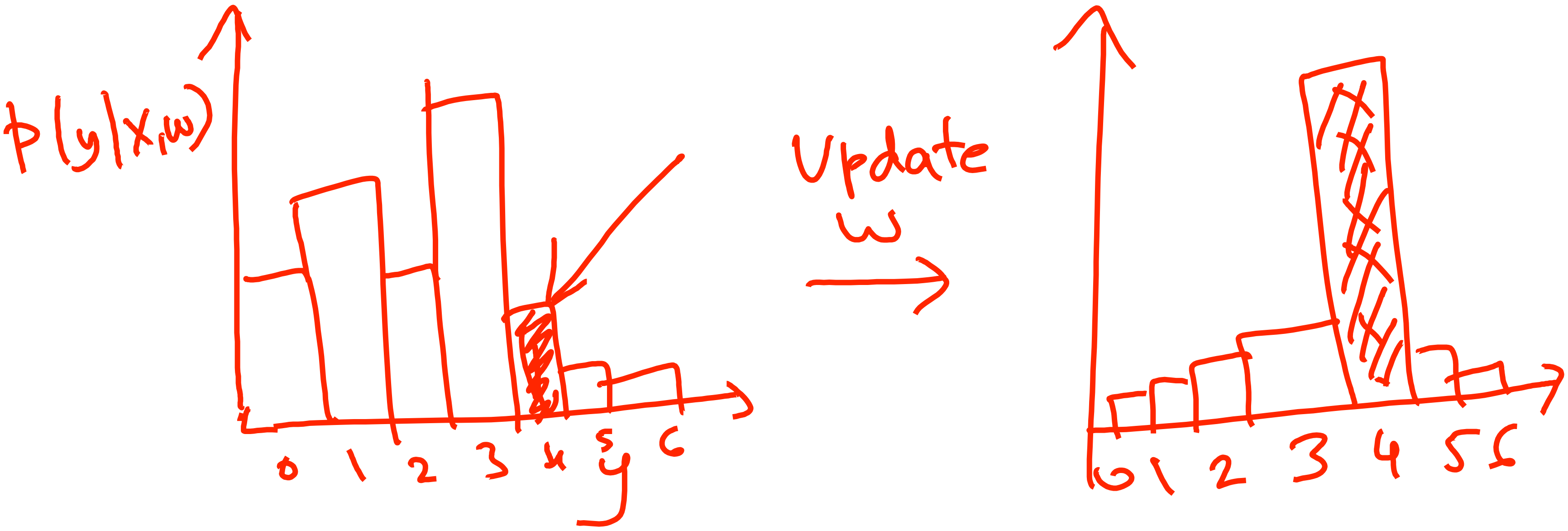
Training MaxEnt Models

- Maximize probability of what is known (training data)
  - make no assumptions about the rest ("maximum entropy")

$$P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

← Make positive

← Normalize



Training MaxEnt Models

- Maximizing the likelihood of the training data incidentally maximizes the entropy (hence "maximum entropy")
- In particular, we maximize conditional log likelihood

$$L(\mathbf{w}) = \mathsf{log} \prod_i \mathsf{P}(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w}) = \sum_i \mathsf{log} \left(\frac{\mathsf{exp}(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i))}{\sum_{\mathbf{y}} \mathsf{exp}(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))} \right)$$

$$= \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \mathsf{log} \sum_{\mathbf{y}} \mathsf{exp}(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

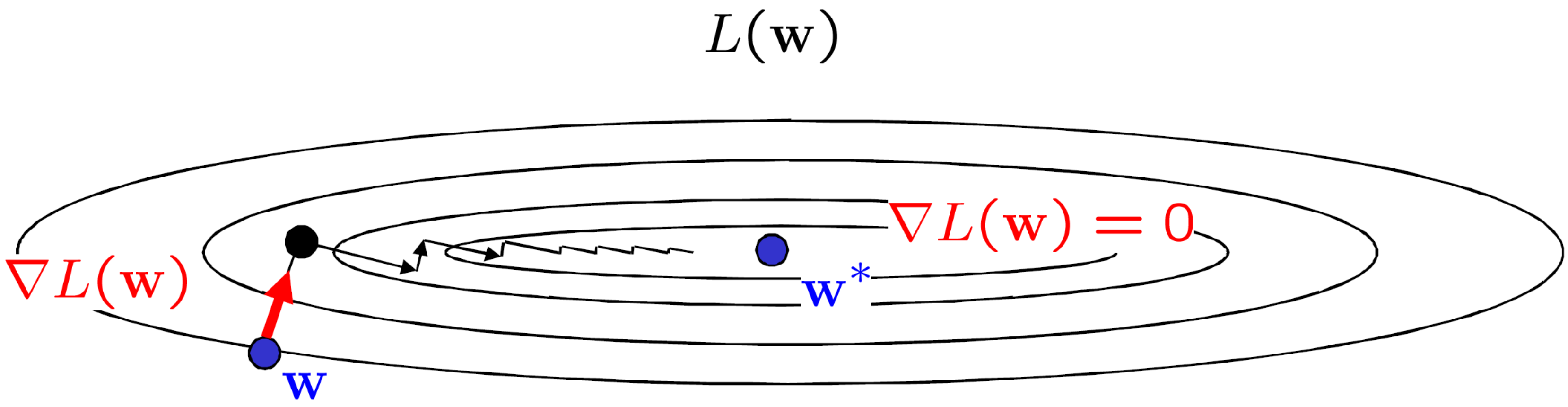
Training MaxEnt Models

- Maximizing the likelihood of the training data incidentally maximizes the entropy (hence “maximum entropy”)
- In particular, we maximize conditional log likelihood

$$L(w) = \log \prod_{(x_i,y_i)} p(y_i \mid x_i, w) = \sum_i \log \left(\frac{\exp w^T f(x_i, y_i)}{\sum_y \exp w^T f(x_i, y)} \right)$$

$$= \sum_i \left(w^T f(x_i, y_i) - \log \sum_y \exp w^T f(x_i, y) \right)$$

Convex Optimization for Training



- The likelihood function is convex. (can get global optimum)
- Many optimization algorithms/software available.
  - Gradient ascent (descent), Conjugate Gradient, L-BFGS, etc
- All we need are:

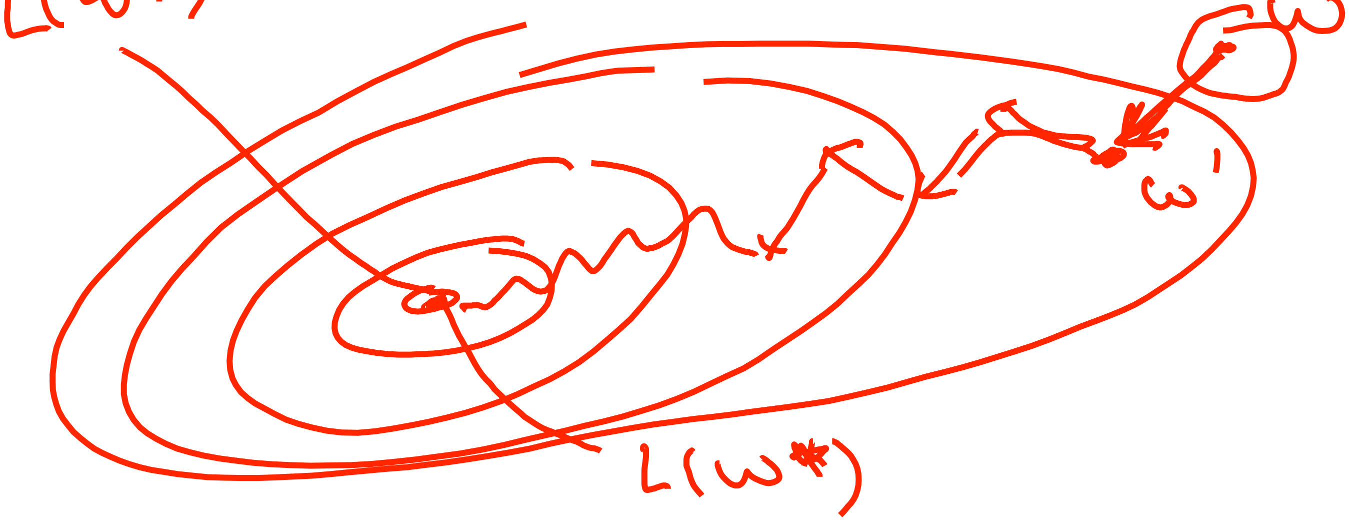
  (1) evaluate the function at current ‘w’

  (2) evaluate its derivative at current ‘w’

Convex Optimization for Training

$\nabla L(w^*) = 0$

$L(w) + \Omega(w) \rightarrow \left[\begin{array}{l} \|w\|_2^2 \\ \|w\|_1 \end{array} \right.$



- The likelihood function is convex. (can get global optimum)
- Many optimization algorithms/software available.
  - Gradient ascent (descent), Conjugate Gradient, L-BFGS, etc

$w_t \leftarrow w_t + \alpha_t \nabla L(w_t)$

- All we need are:

  (1) evaluate the function at current 'w' – $L(w)$

  (2) evaluate its derivative at current 'w' – $\nabla L(w)$

Training MaxEnt Models

$$L(\mathbf{w}) = \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n \right)$$

Total count of feature n in correct candidates

Expected count of feature n in predicted candidates

Training with Regularization

$$L(\mathbf{w}) = -k||\mathbf{w}||^2 + \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = -2k\mathbf{w}_n + \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n \right)$$

Big weights are bad

Expected count of feature n in predicted candidates

Total count of feature n in correct candidates

Training MaxEnt Models

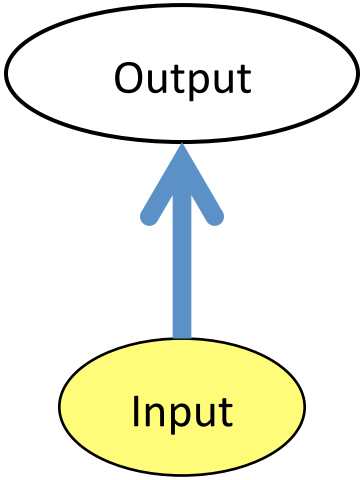
$$L(w) = \sum_i \underline{w^T f(x_i, y_i)} - \log \sum_y \exp w^T f(x_i, y)$$

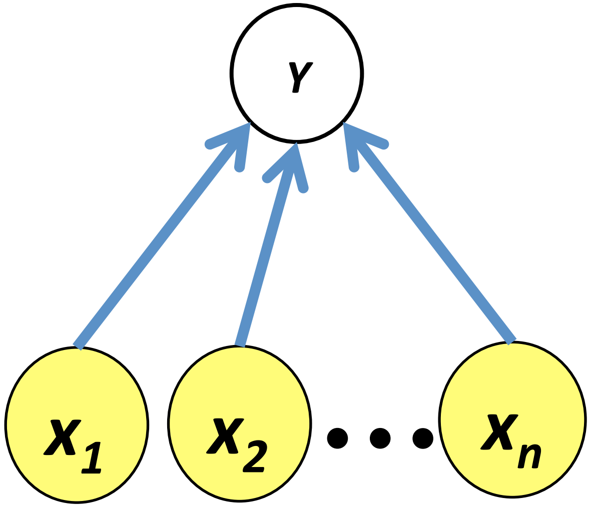
$$\frac{\nabla L(w)}{\partial w_n} = \sum_i \underbrace{f_n(x_i, y_i)}_{\text{Counting feature n in the data}} - \sum_y \underbrace{p(y|x_i) f(x_i, y)_n}_{\text{Expected count of feature n in predictions}}$$

Training with Regularization

Graphical Representation of MaxEnt

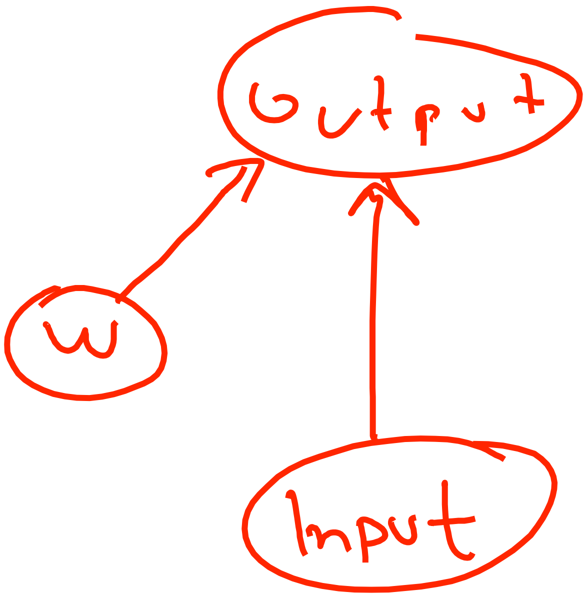
$$\mathsf{P}(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

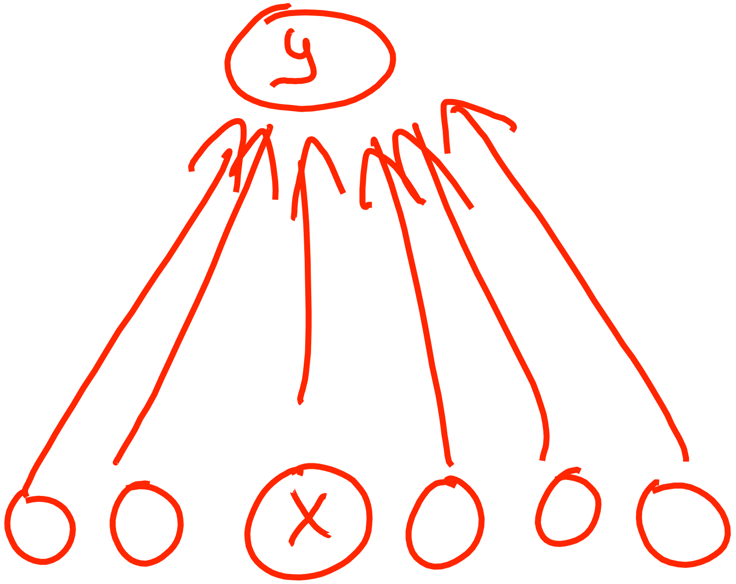




Graphical Representation of MaxEnt

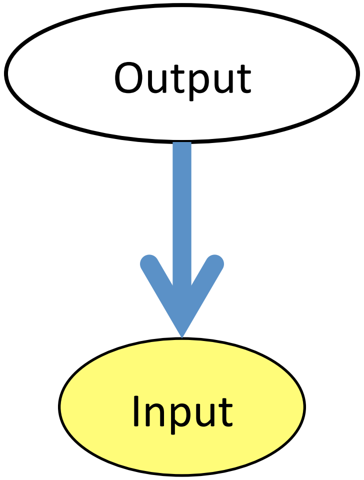
$$\mathsf{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

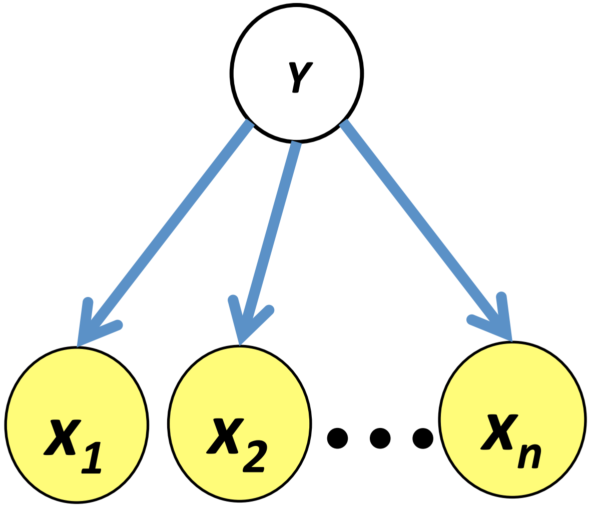




Graphical Representation of **Naïve Bayes**

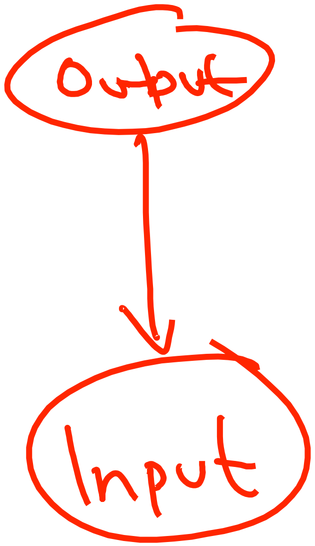
$$P(X\,|\,Y)=\prod_{j=1}P(x_j\,|\,Y)$$

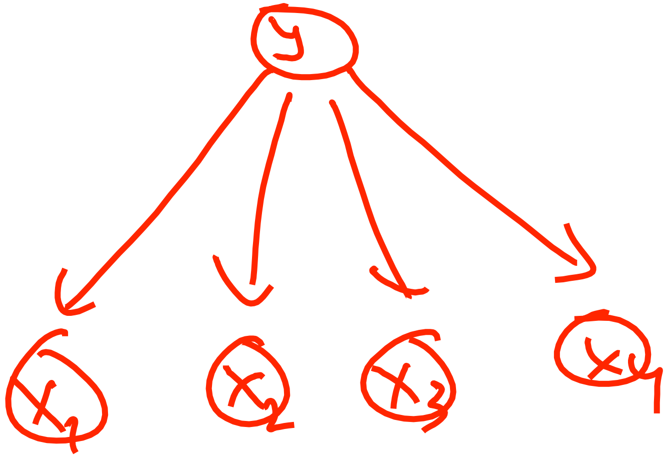




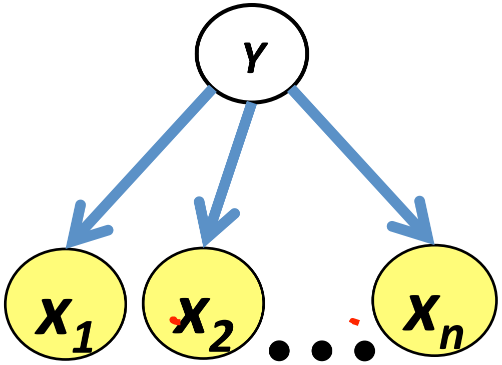
Graphical Representation of **Naïve Bayes**

$$P(X \mid Y) = \prod_{j=1} P(x_j \mid Y)$$

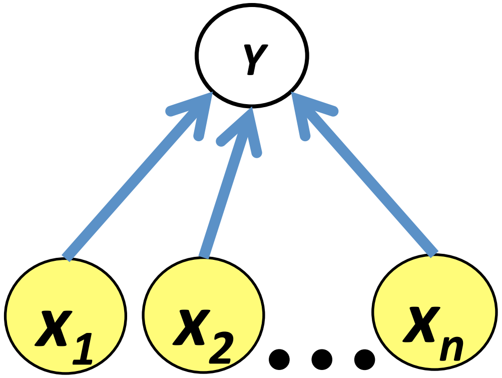




Naïve Bayes



MaxEnt



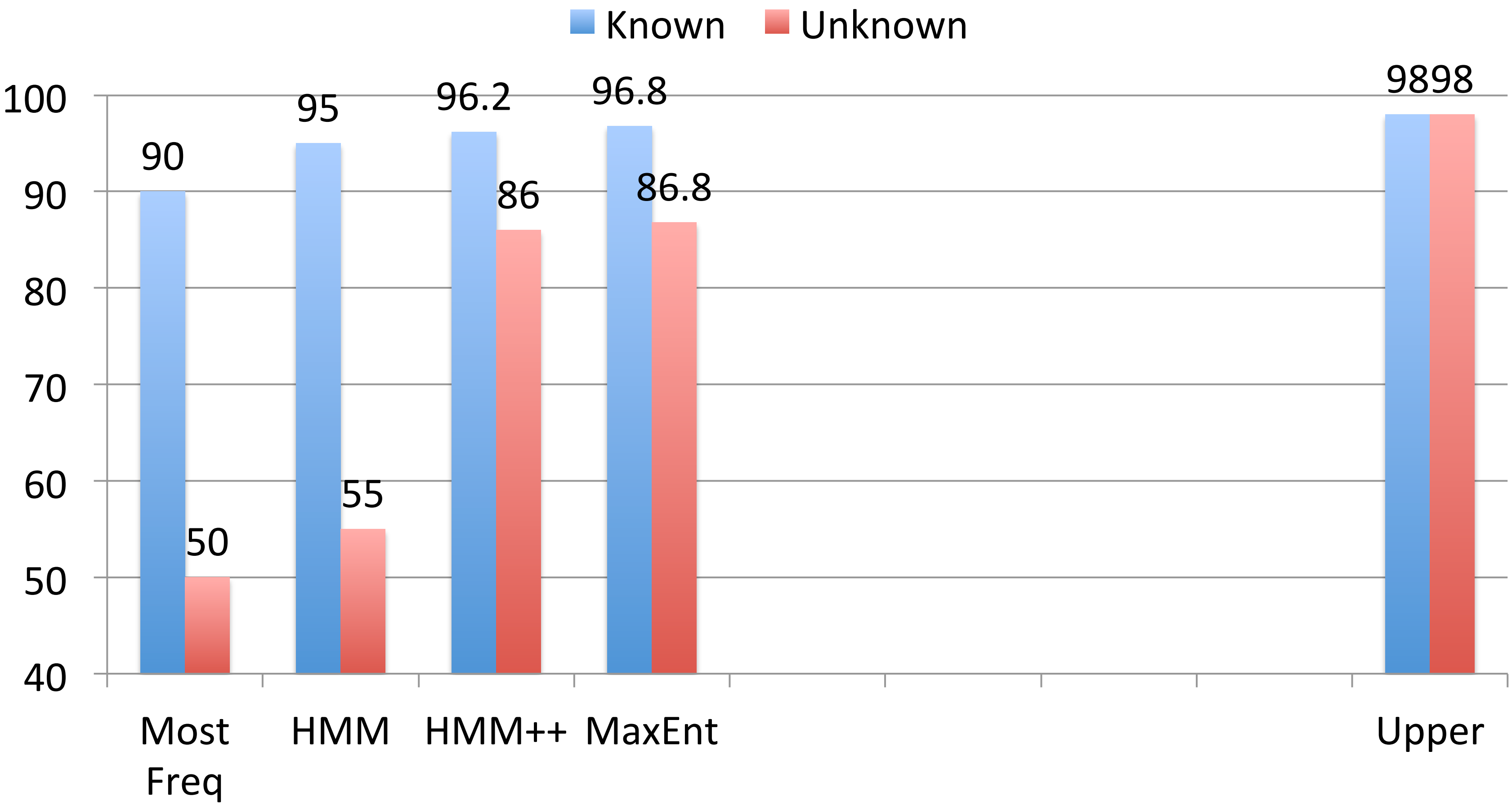
Naïve Bayes Classifier	Maximum Entropy Classifier
"***Generative***" models ➔ p(input | output) ➔ For instance, for text categorization, P(words | category) ➔ Unnecessary efforts on generating input	"***Discriminative***" models ➔ p(output | input) ➔ For instance, for text categorization, P(category | words) ➔ Focus directly on predicting the output
➔ Independent assumption among input variables: Given the category, each word is generated independently from other words (too strong assumption in reality!) ➔ Cannot incorporate arbitrary/redundant/ overlapping features	➔ By conditioning on the entire input, we don't need to worry about the independent assumption among input variables ➔ Can incorporate arbitrary features: redundant and overlapping features

Overview: POS tagging Accuracies

- Roadmap of (known / unknown) accuracies:

– Most freq tag:	~90% / ~50%
– Trigram HMM:	~95% / ~55%
– TnT (HMM++):	96.2% / 86.0%
– Maxent $P(s_i\|x)$:	96.8% / 86.8%
– Upper bound:	~98%

POS Results



Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
96.2
86
96.8
86.8
9898
Most Freq
HMM
HMM++
MaxEnt
Upper

Outline

- POS Tagging
- MaxEnt
- MEMM - Max Entropy Markov Model
- CRFs
- Wrap-up
- Optional: Perceptron

Sequence Modeling

- Predicted POS of neighbors is important

The ___ N Red ___

A ___ N Green ___

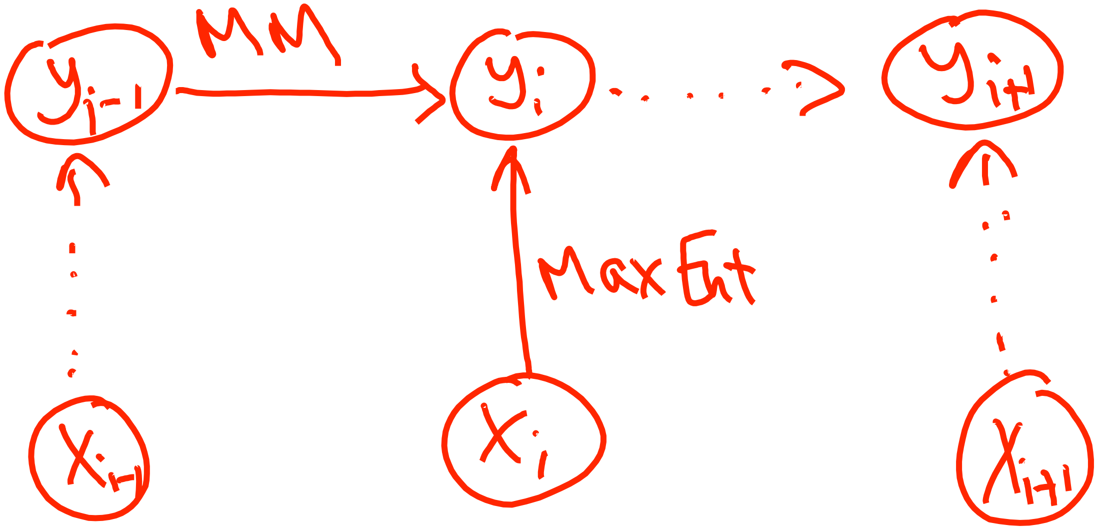
An ___ N Big ___

⋮

DT ___ JJ ___

MEMM Taggers

- One step up: also condition on previous tags



MEMM Taggers

- Conditioning on previous tags

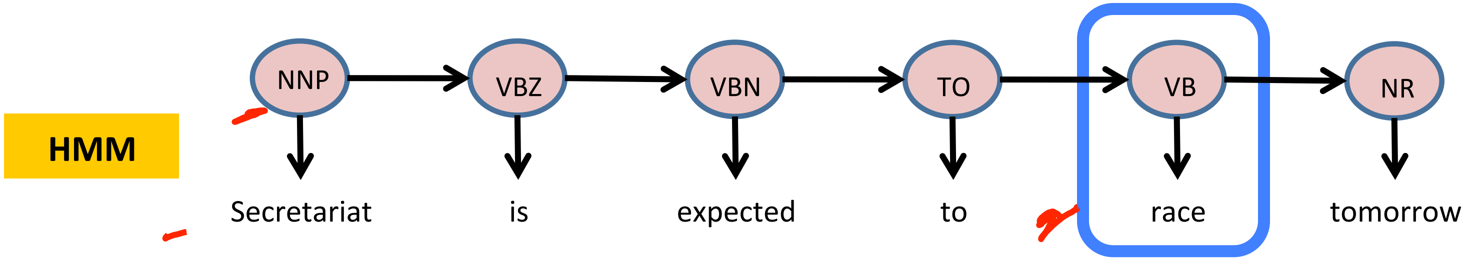
$$\underline{p(s_1 \dots s_m | x_1 \dots x_m)} = \prod_{i=1}^{m} \underline{p(s_i | s_1 \dots s_{i-1}, x_1 \dots x_m)}$$

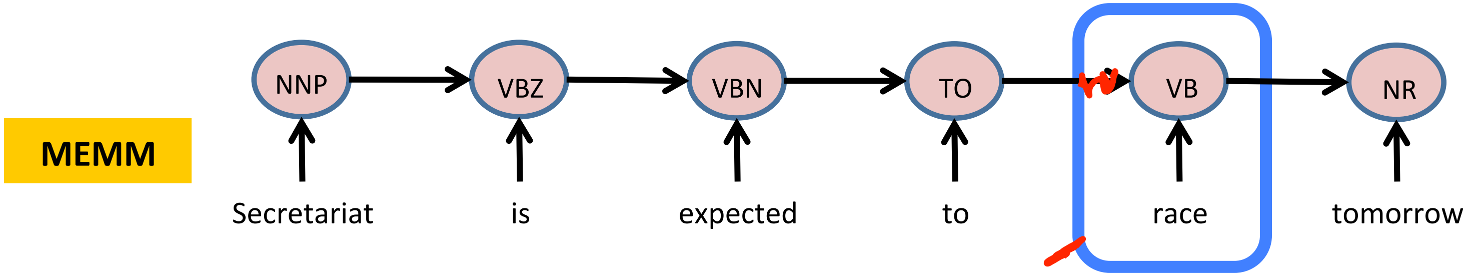
$$= \prod_{i=1}^{m} \underline{p(s_i | s_{i-1}, x_1 \dots x_m)}$$

  - Train up $p(s_i|s_{i-1},x_{1..}.x_m)$ as a discrete log-linear (maxent) model, then use to score sequences

$$\underline{p(s_i | s_{i-1}, x_1 \dots x_m)} = \frac{\exp\left(w \cdot \boxed{\phi}(x_1 \dots x_m, i, s_{i-1}, s_i)\right)}{\sum_{s'} \exp\left(w \cdot \phi(x_1 \dots x_m, i, s_{i-1}, s')\right)}$$

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

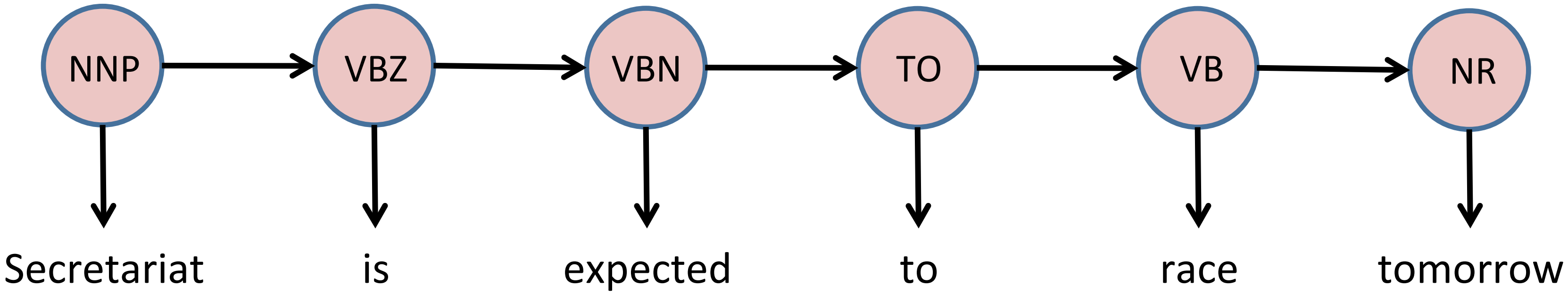




HMM	MEMM
"Generative" models ➔ joint probability **p(words, tags)** ➔"generate" input (in addition to tags) ➔ but we need to predict tags, not words!	"Discriminative" or "Conditional" models ➔ conditional probability **p(tags | words)** ➔"condition" on input ➔ Focusing only on predicting tags
Probability of each slice = emission * transition = p(word_i | tag_i) * p(tag_i | tag_i-1) = ➔ Cannot incorporate long distance features	Probability of each slice = p(tag_i | tag_i-1, word_i) or p(tag_i | tag_i-1, all words) ➔ Can incorporate long distance features

HMM v.s. MEMM

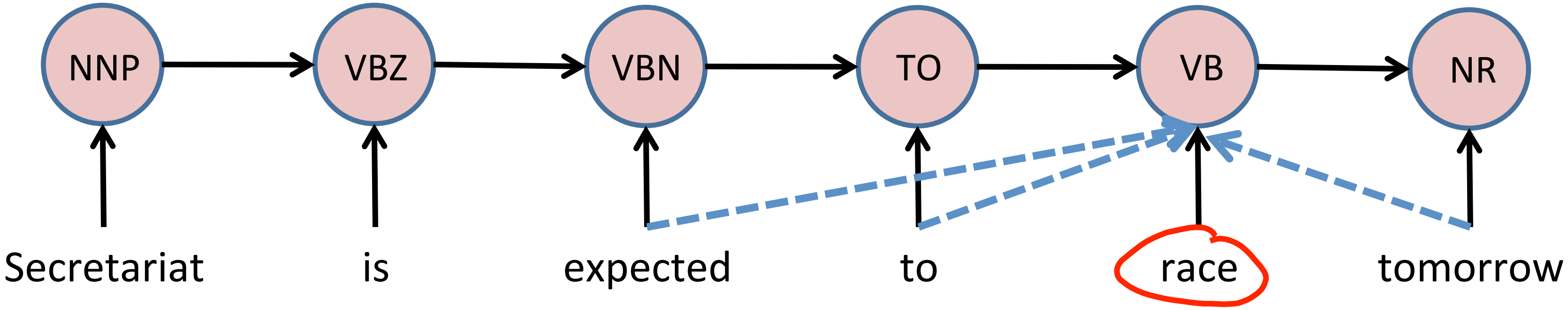
HMM



NNP VBZ VBN TO VB NR

Secretariat is expected to race tomorrow

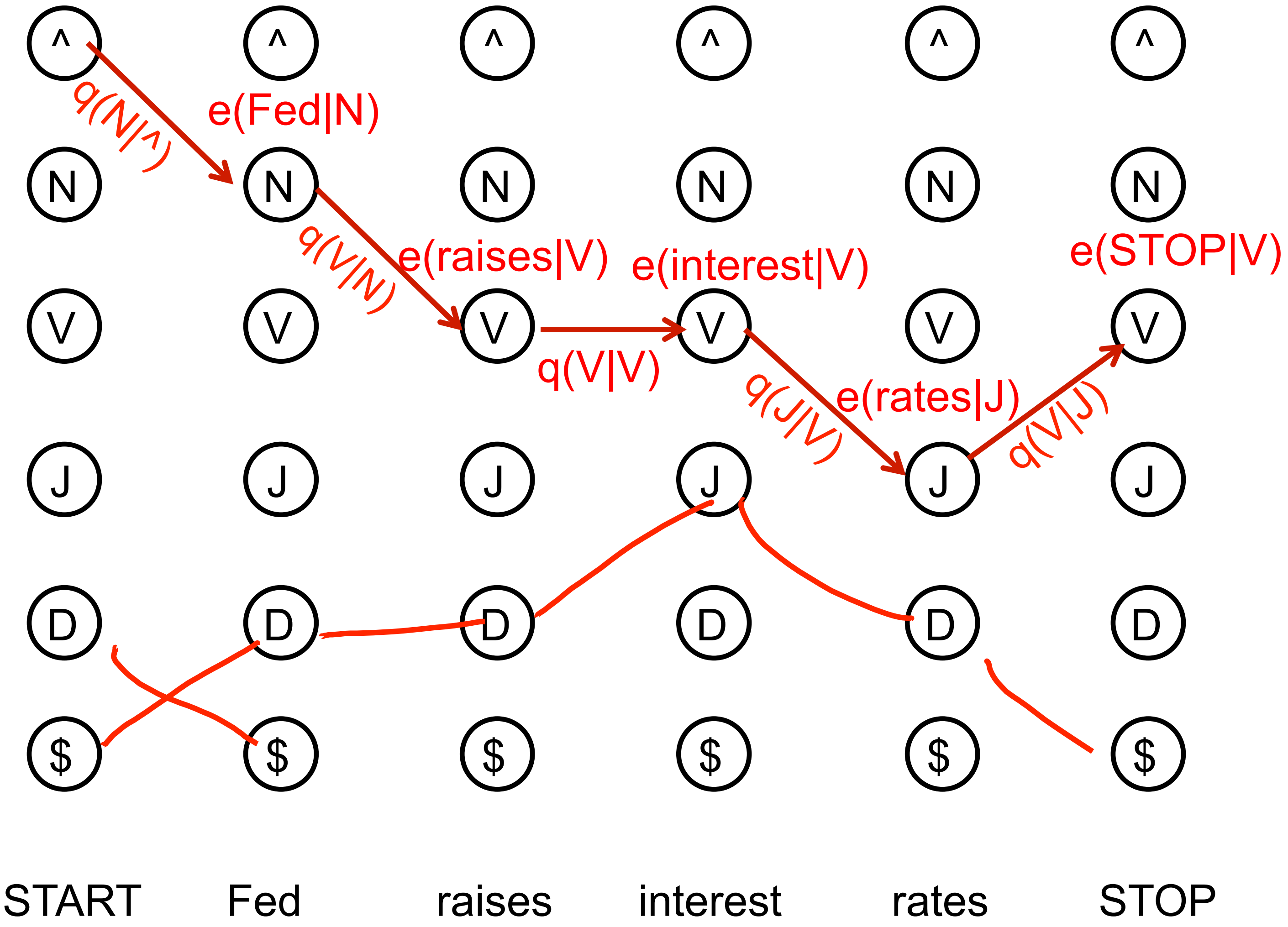
MEMM



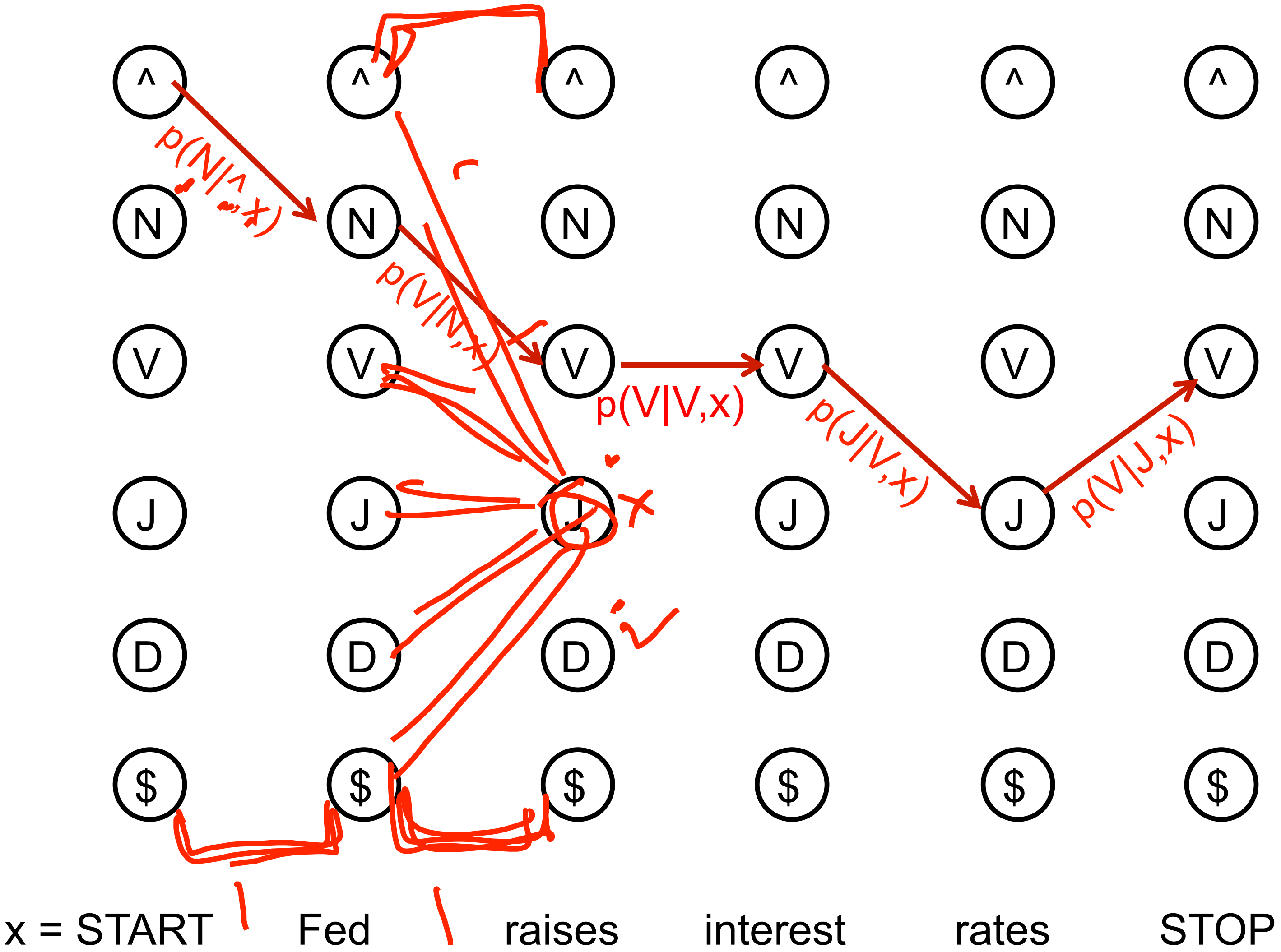
NNP VBZ VBN TO VB NR

Secretariat is expected to race tomorrow

The HMM State Lattice / Trellis (repeat slide)



The MEMM State Lattice / Trellis



Decoding: $p(s_1 \ldots s_m | x_1 \ldots x_m) = \prod_{i=1}^{m} p(s_i | s_1 \ldots s_{i-1}, x_1 \ldots x_m)$

- Decoding maxent taggers:
  - Just like decoding HMMs
  - Viterbi, beam search, posterior decoding

- Viterbi algorithm (HMMs):
  - Define $\pi(i,s_i)$ to be the max score of a sequence of length i ending in tag s_i

$$\pi(i, s_i) = \max_{s_{i-1}} \underline{e(x_i | s_i)}\,\underline{q(s_i | s_{i-1})}\,\pi(i-1, s_{i-1})$$

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

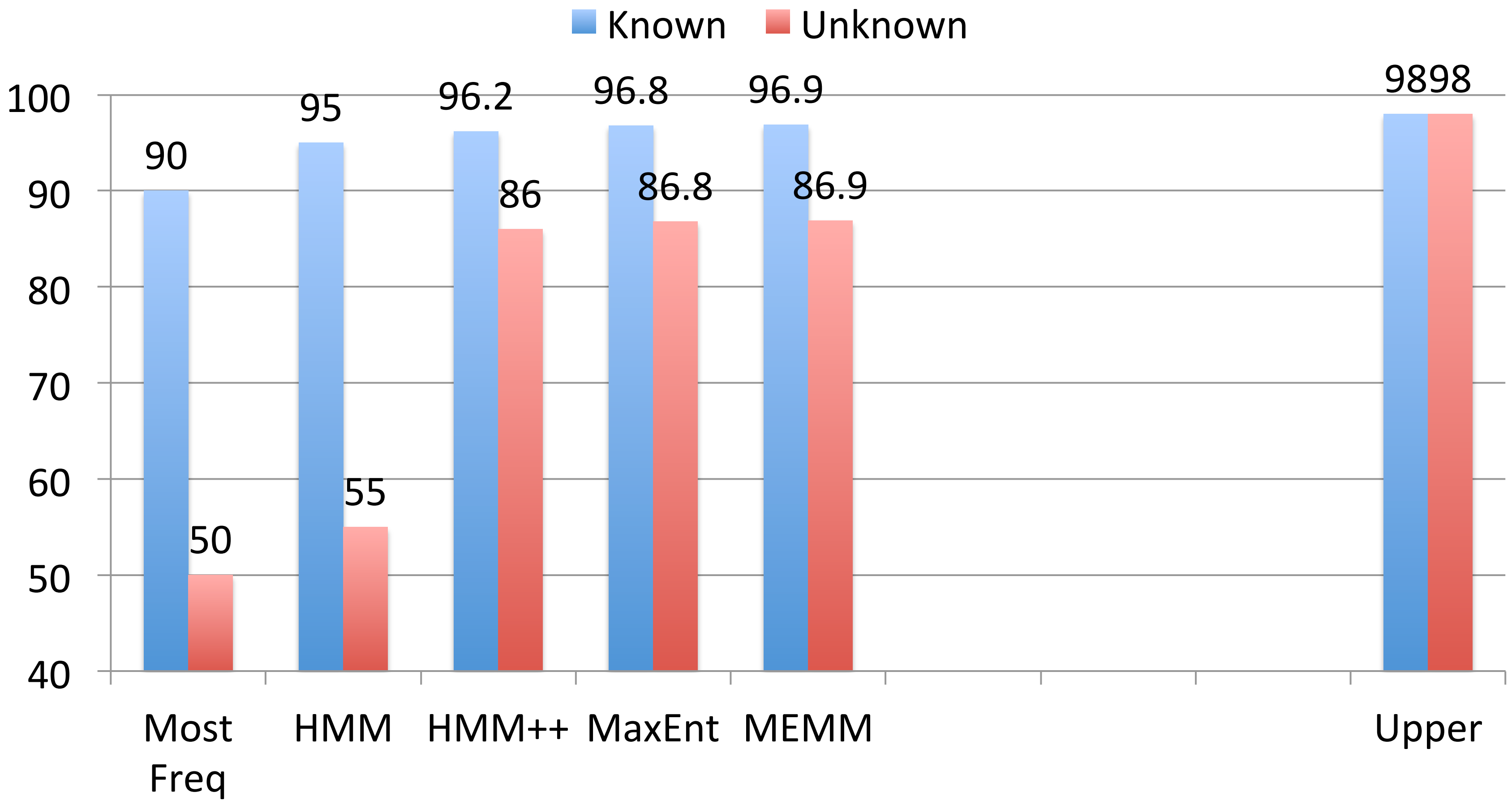
$$\pi(i, s_i) = \max_{s_{i-1}} \underline{p(s_i | s_{i-1}, x_1 \ldots x_m)}\,\pi(i-1, s_{i-1})$$

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:

Most freq tag:	~90% / ~50%
Trigram HMM:	~95% / ~55%
TnT (HMM++):	96.2% / 86.0%
Maxent $P(s_i\|x)$:	96.8% / 86.8%
MEMM tagger:	96.9% / 86.9%
Upper bound:	~98%

POS Results

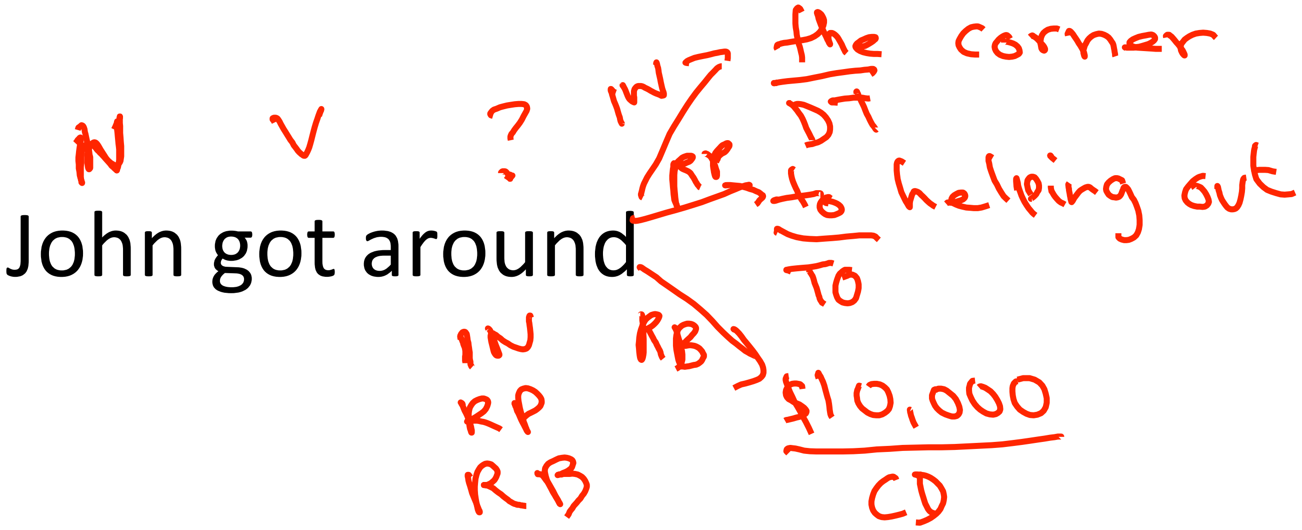

Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
96.2
86
96.8
86.8
96.9
86.9
9898
Most Freq
HMM
HMM++
MaxEnt
MEMM
Upper

Outline

- POS Tagging
- MaxEnt
- MEMM
- CRFs
- Wrap-up
- Optional: Perceptron

Global Sequence Modeling

- MEMM and MaxEnt are "local" classifiers
  - MaxEnt more so that MEMM
  - make decision conditioned on local information
  - Not much of a "flow" of information



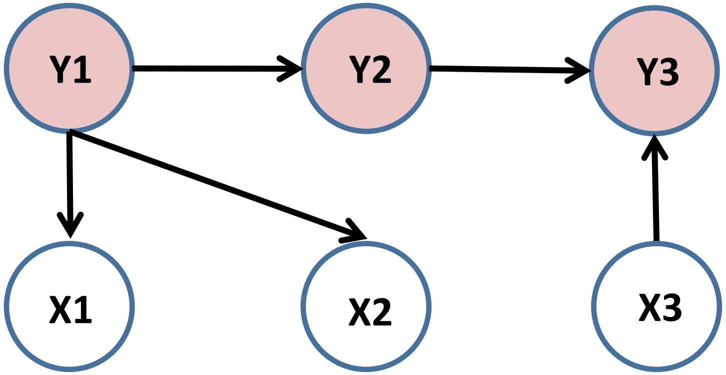
John got around

- Make prediction on the whole chain directly!

Global Discriminative Taggers

- Newer, higher-powered discriminative sequence models
  - CRFs (also perceptrons, M3Ns)
  - Do not decompose training into independent local regions
  - Can be slower to train: repeated inference during training set
- However: one issue worth knowing about in local models
  - “**Label bias**” and other **explaining away effects**
  - MEMM taggers’ local scores can be near one without having both good “transitions” and “emissions”
  - This means that often evidence doesn’t flow properly
  - Also: in decoding, condition on predicted, not gold, histories

Graphical Models

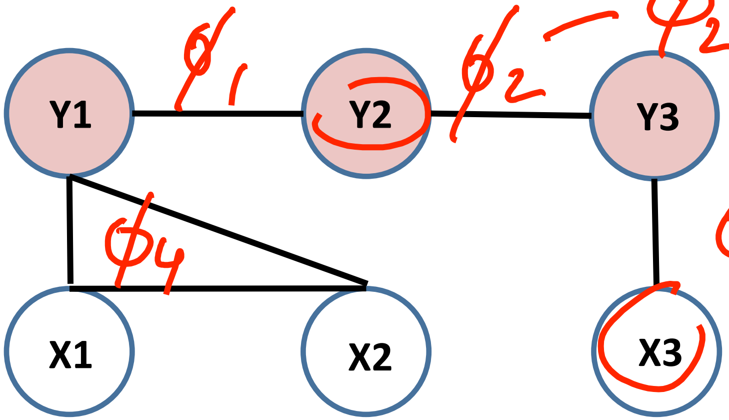


- Conditional probability for each node
  - e.g. p(Y3 | Y2, X3) for Y3
  - e.g. p(X3) for X3
- Conditional independence
  - e.g. p(Y3 | Y2, X3) = p(Y3 | Y1, Y2, X1, X2, X3)
- Joint probability of the entire graph

  = product of conditional probability of each node

$$P(Y_s, X_s) = P(X_3)\, P(Y_1)\, P(Y_2 | Y_1)\, P(X_1 | Y_1)\, P(X_2 | Y_1)\, P(Y_3 | X_3, Y_2)$$

Undirected Graphical Model Basics



$\phi_2(Y_2, Y_3) \in R^+$

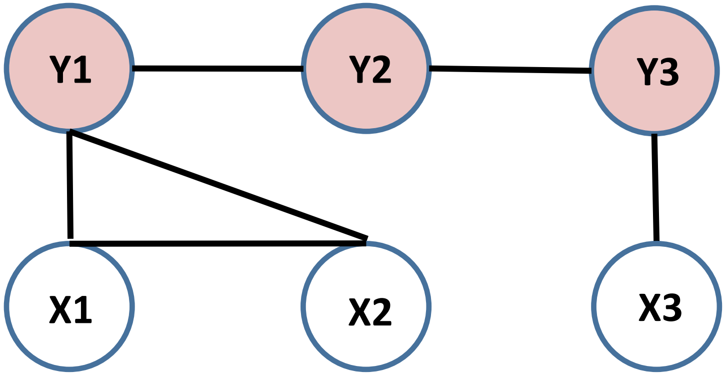
$\phi_3(Y_3, X_3) \in R^+$

- Conditional independence
  - e.g. p(Y3 | all other nodes) = p(Y3 | Y3' neighbor)
- No conditional probability for each node
- Instead, "***potential function***" for each ***clique***
  - e.g. $\boxed{?}$ (X1, X2, Y1) or $\boxed{?}$ (Y1, Y2)
- Typically, log-linear potential functions

➔ $\underline{\boxed{?}\text{ (Y1, Y2)}}$ = $\underline{\text{exp } \boxed{?}_k \ w_k f_k \text{ (Y1, Y2)}}$

$$P(Y_s, X_s) \propto \phi_1(Y_1, Y_2)$$
$$\phi_2(Y_2, Y_3)$$
$$\vdots$$

Undirected Graphical Model Basics



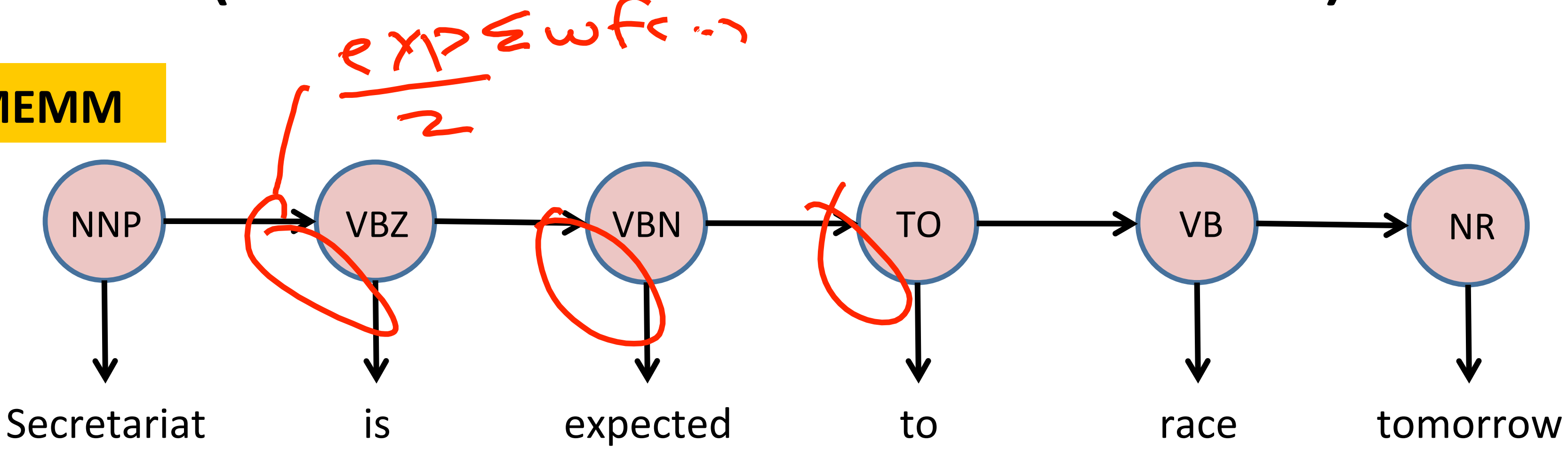
- Joint probability of the entire graph

$$\underline{P(\vec{Y})} = \frac{1}{\boxed{Z}} \prod_{\text{clique } C} \underline{\varphi(\vec{Y}_C)} = \frac{1}{Z} \prod_{c} \exp \sum w_k^c f_k^c(\ldots)$$

$$Z = \sum_{\underline{\vec{Y}}} \prod_{\text{clique } C} \varphi(\vec{Y}_C)$$

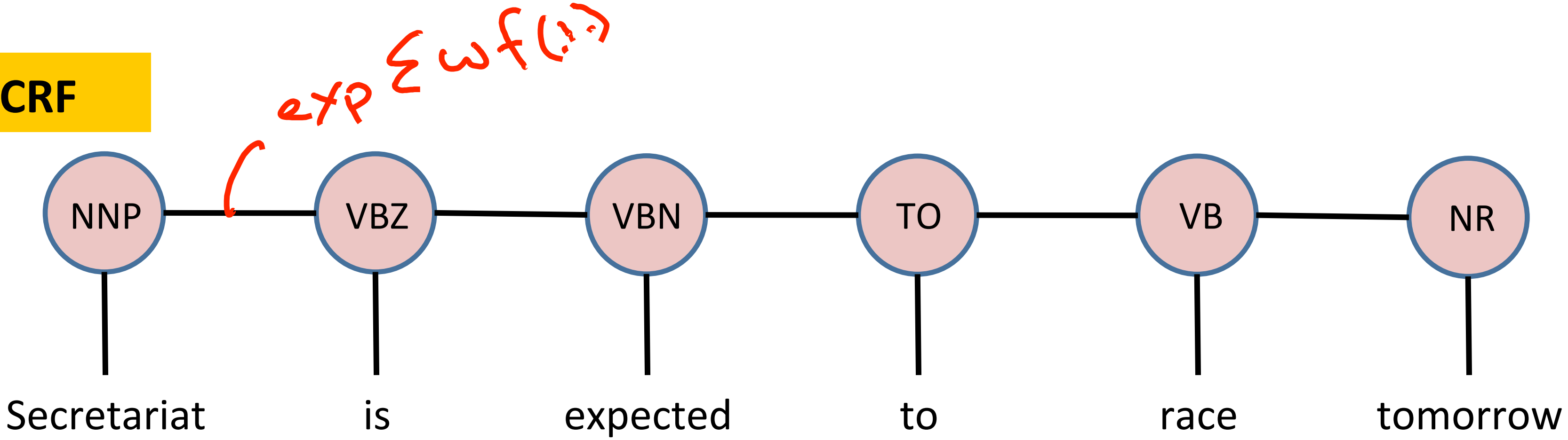
MEMM v.s. CRF
(Conditional Random Fields)

MEMM



$$\frac{\exp \sum w f(\cdot\cdot)}{Z}$$

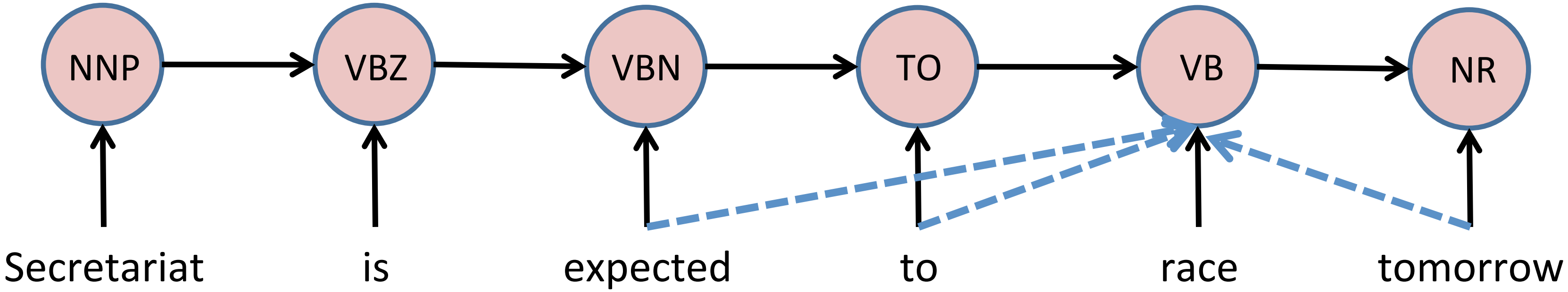
CRF



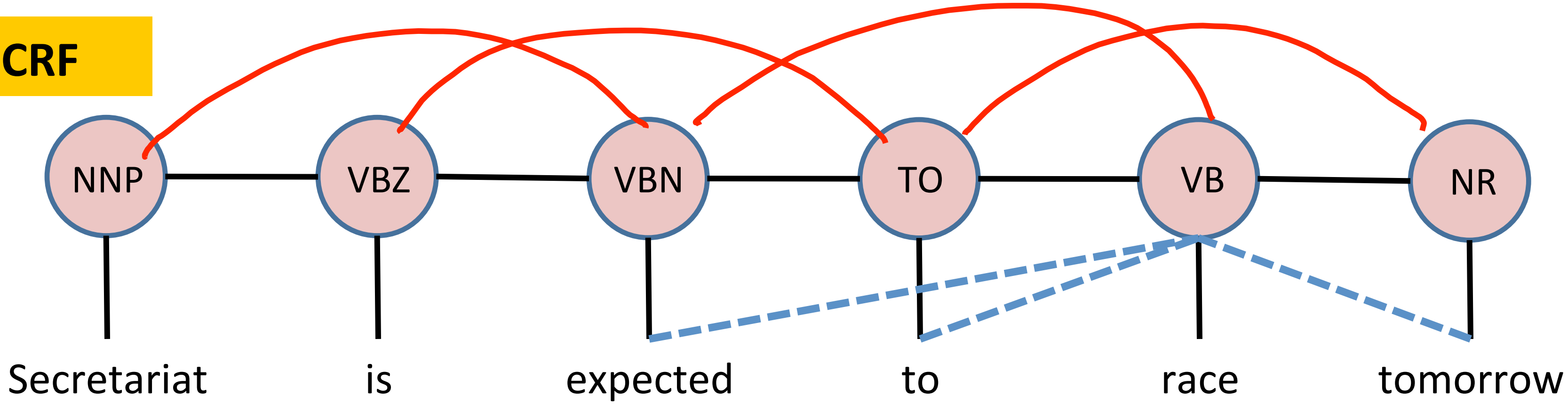
$$\exp \sum w f(\cdot\cdot)$$

MEMM v.s. CRF

MEMM



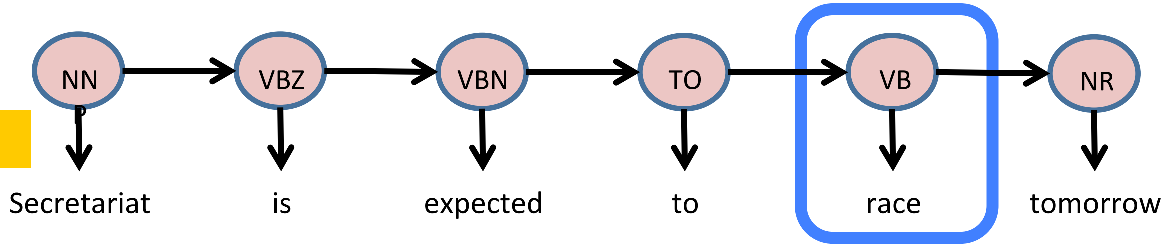
CRF



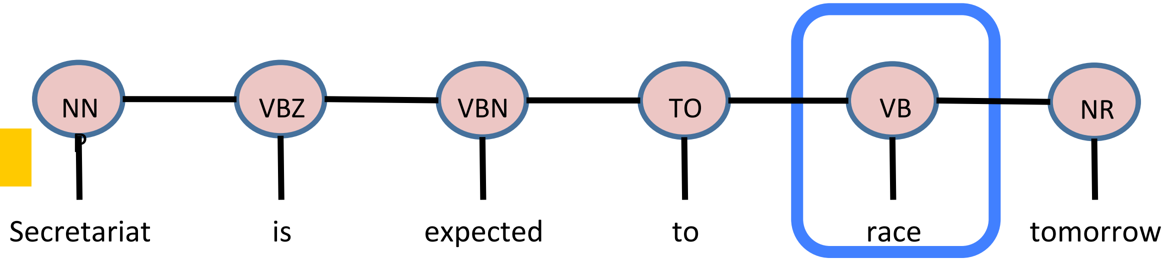
MALLET CRF++ nlptk

FACTORIE

MEMM



CRF



MEMM	CRF
Directed graphical model	Undirected graphical model
“Discriminative” or “Conditional” models ➔ conditional probability **p(tags | <u>words</u>)**	
Probability is defined for each slice = P (tag_i | tag_i-1, word_i) or p (tag_i | tag_i-1, all words)	Instead of probability, ***potential (energy function)*** is defined for each slide = ⍰ (tag_i, tag_i-1) * ⍰ (tag_i, word_i) or f(tag_i, tag_i-1, all words) * ⍰ (tag_i, all words)
➔ Can incorporate long distance features	

Conditional Random Fields (CRFs)

- Maximum entropy (logistic regression)

Sentence: $x=x_1 \ldots x_m$

$$p(y|x;w) = \frac{\exp\left(w \cdot \phi(x,y)\right)}{\sum_{y'} \exp\left(w \cdot \phi(x,y')\right)}$$

Tag Sequence: $y=s_1 \ldots s_m$

  - Learning: maximize the (log) conditional likelihood of training data $\{(x_i, y_i)\}_{i=1}^{n}$

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{n} \left(\phi_j(x_i, y_i) - \sum_{y} p(y|x_i;w)\phi_j(x_i,y) \right) - \lambda w_j$$

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

[Lafferty, McCallum, Pereira 01]

Conditional Random Fields (CRFs)

- Maximum entropy (logistic regression)

$$p(\underset{\text{seq}}{y}|\underset{\text{seq}}{x};w) = \frac{\exp\left(w\cdot\phi(x,y)\right)}{\sum_{\boxed{y'}}\exp\left(w\cdot\phi(x,y')\right)}$$

- Learning: maximize (log) conditional likelihood of training data $\{(x_i,y_i)\}_{i=1}^{n}$

$$\frac{\partial}{\partial \underbrace{w_j}_{}}\underline{L(w)} = \sum_{i=1}^{n}\left(\phi_j(x_i,y_i) - \underbrace{\sum_{y}\underline{p(y|x_i;w)}\phi_j(x_i,y)}_{}\right)$$

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

[Lafferty, McCallum, Pereira 01]

Decoding

$$s^* = \arg\max_s p(s|x;w)$$

- CRFs
  - Features must be local, for x=$x_1 \ldots x_m$, and s=$s_1 \ldots s_m$

$$p(s|x;w) = \frac{\exp\left(w \cdot \Phi(x,s)\right)}{\sum_{s'} \exp\left(w \cdot \Phi(x,s')\right)} \quad \Phi(x,s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j)$$

$$\arg\max_s \frac{\exp\left(w \cdot \Phi(x,s)\right)}{\sum_{s'} \exp\left(w \cdot \Phi(x,s')\right)} = \arg\max_s \exp\left(w \cdot \Phi(x,s)\right)$$

$$= \arg\max_s w \cdot \Phi(x,s)$$

- Same as Linear Perceptron!!!

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

Decoding

$$\underline{s^*} = \underline{\arg\max_s p(s|x;w)}$$

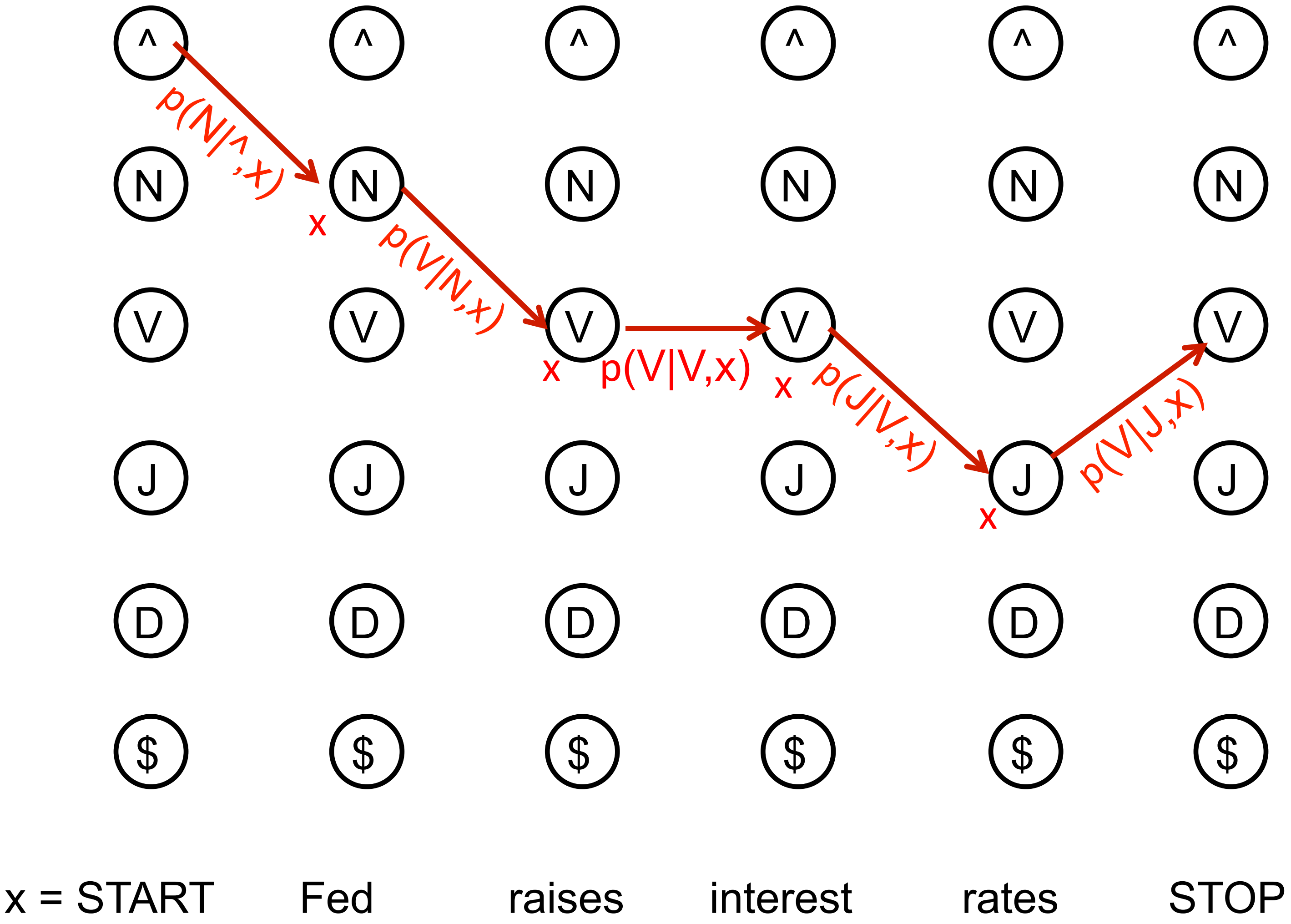
- CRFs
  - Features must be local, for x=x$_1$…x$_m$, and s=s$_1$…s$_m$

$$\underline{p(s|x;w)} = \left(\frac{\exp\left(w \cdot \boxed{\Phi}(x,s)\right)}{\sum_{s'} \exp\left(w \cdot \Phi(x,s')\right)} \right] \qquad \Phi(x,s) = \sum_{j=1}^{m} \phi(x, j, \underline{s_{j-1}, s_j})$$

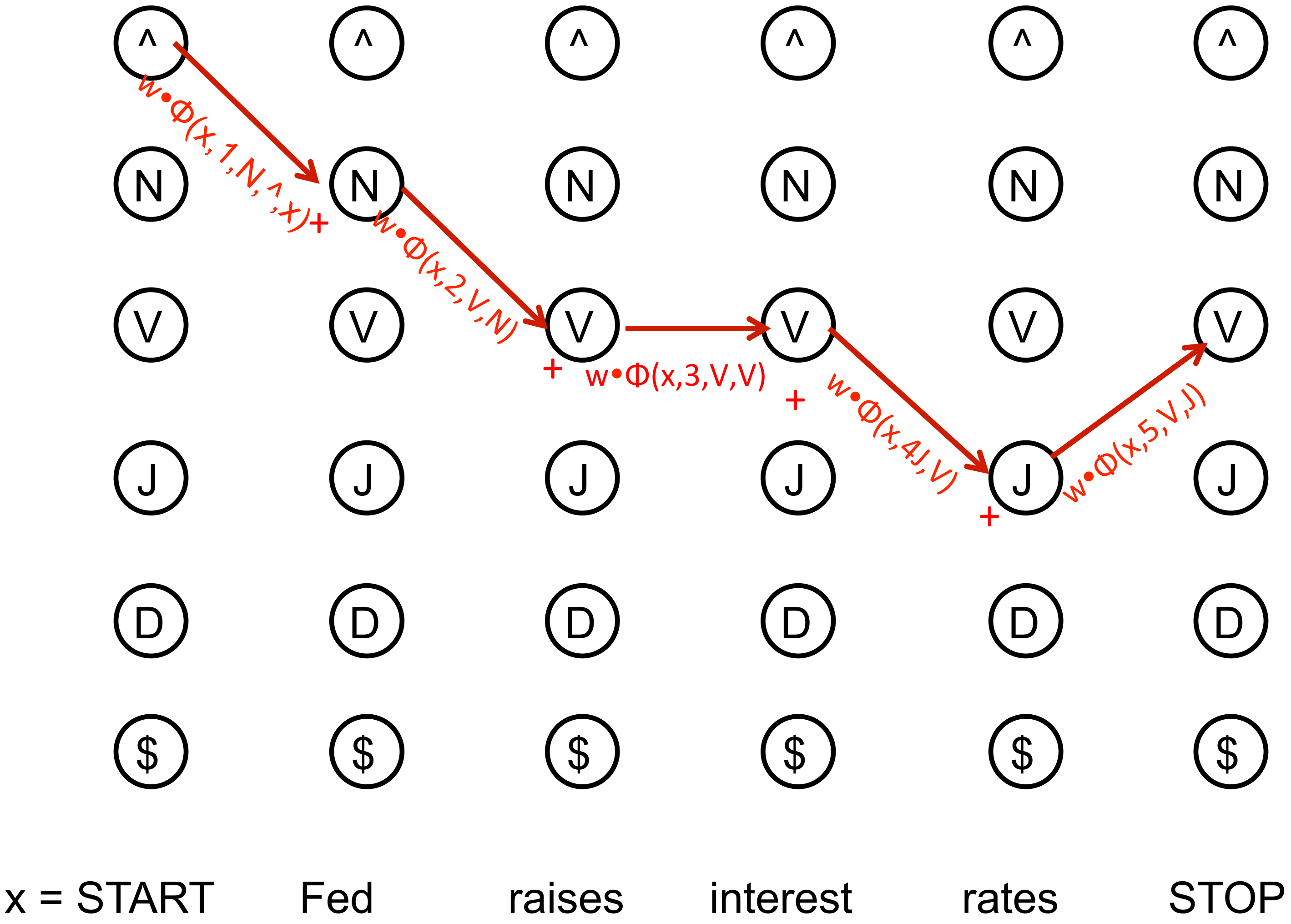
$$\arg\max_s \frac{\exp\left(w \cdot \Phi(x,s)\right)}{\cancel{\sum_{s'} \exp\left(w \cdot \Phi(x,s')\right)}} = \arg\max_s \cancel{\exp}\left(w \cdot \Phi(x,s)\right)$$

$$= \arg\max_s w \cdot \Phi(x,s)$$

The MEMM State Lattice / Trellis (repeat)



CRF State Lattice / Trellis



CRFs: Computing Normalization*

$$p(s|x;w) = \frac{\exp\left(w \cdot \Phi(x,s)\right)}{\sum_{s'} \exp\left(w \cdot \Phi(x,s')\right)} \qquad \Phi(x,s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j)$$

$$\sum_{\underline{s'}} \exp\left(w \cdot \Phi(x,s')\right) = \sum_{s'} \exp\left(\sum_{\boxed{j}} w \cdot \phi\underline{(x, j, s_{j-1}, s_j)}\right)$$

$$= \sum_{s'} \prod_{j} \exp\left(w \cdot \phi(x, j, s_{j-1}, s_j)\right)$$

Define $\underline{\text{norm}(i,s_i)}$ to sum of scores for sequences ending in position i

$$norm(i, y_i) = \sum_{s_{i-1}} \underline{\exp\left(w \cdot \phi(x, i, s_{i-1}, s_i)\right)}\, \underline{norm(i-1, s_{i-1})}$$

- Forward Algorithm! Remember HMM case:

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

CRFs: Computing Gradient*

$$p(s|x;w) = \frac{\exp\left(w \cdot \Phi(x,s)\right)}{\sum_{s'} \exp\left(w \cdot \Phi(x,s')\right)} \qquad \Phi(x,s) = \sum_{j=1}^{m} \phi(x,j,s_{j-1},s_j)$$

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{n} \left(\Phi_j(x_i,s_i) - \sum_{s} p(s|x_i;w)\Phi_j(x_i,s) \right) + ||w||_2^2$$

$$\sum_{s} p(s|x_i;w)\Phi_j(x_i,s) = \sum_{s} p(s|x_i;w) \sum_{j=1}^{m} \phi_k(x_i,j,s_{j-1},s_j)$$

$$= \sum_{j=1}^{m} \sum_{a,b} \sum_{s:s_{j-1}=a,s_b=b} p(s|x_i;w)\phi_k(x_i,j,s_{j-1},s_j)$$

- Need forward and backward messages

See notes for full details!

$\alpha_i = \alpha_{i-1}$

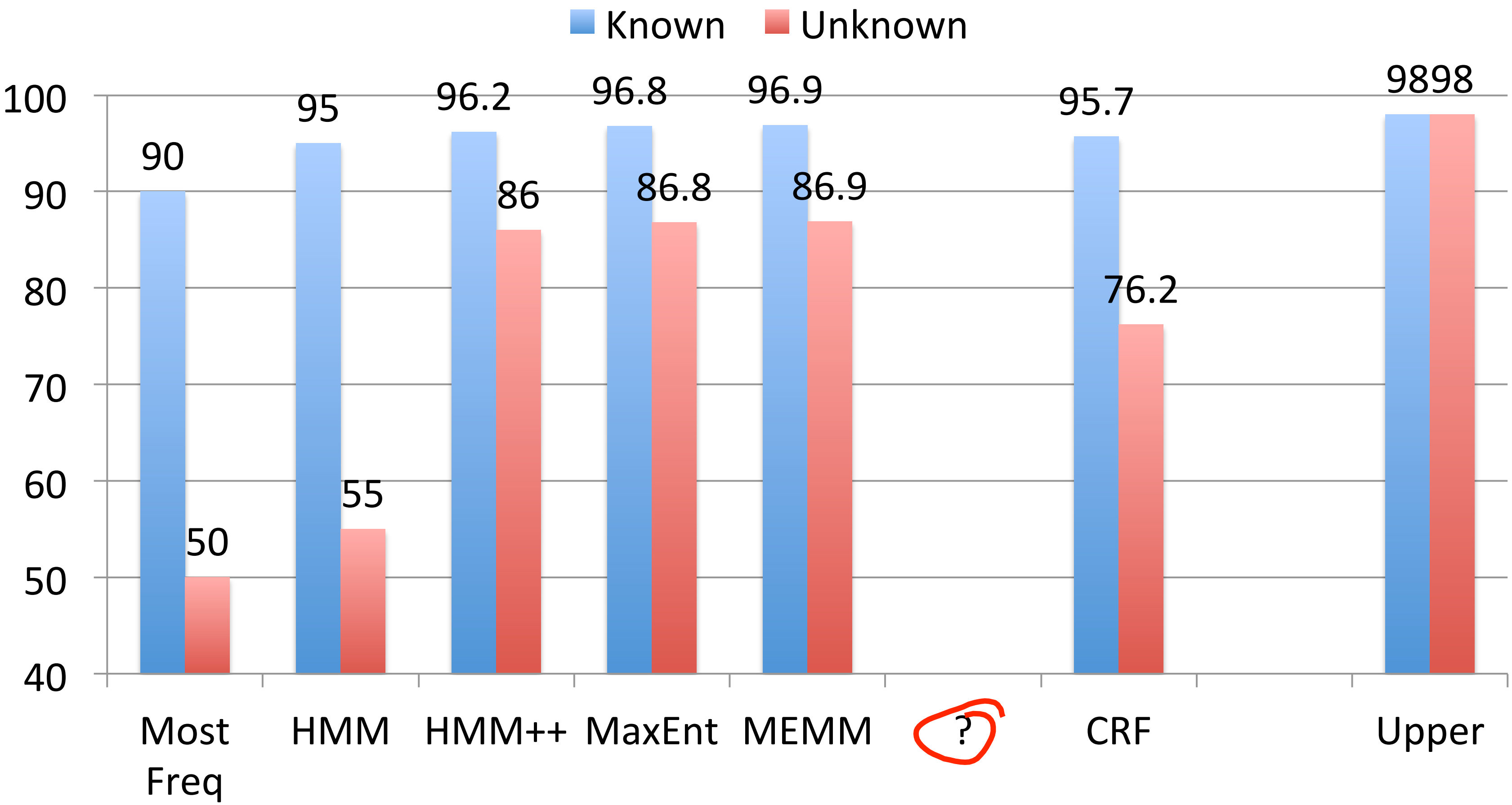
$\beta_i = \beta_{i+1}$

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:

– Most freq tag:	~90% / ~50%
– Trigram HMM:	~95% / ~55%
– TnT (HMM++):	96.2% / 86.0%
– Maxent $P(s_i\|x)$:	96.8% / 86.8%
– MEMM tagger:	96.9% / 86.9%
– CRF (untuned)	95.7% / 76.2%
– Upper bound:	~98%

POS Results



Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
96.2
86
96.8
86.8
96.9
86.9
95.7
76.2
9898
Most Freq
HMM
HMM++
MaxEnt
MEMM
?
CRF
Upper

Cyclic Network [Toutanova et al 03]

- Train two MEMMs, multiple together to score
- And be very careful
    - Tune regularization
    - Try lots of different features
    - See paper for full details

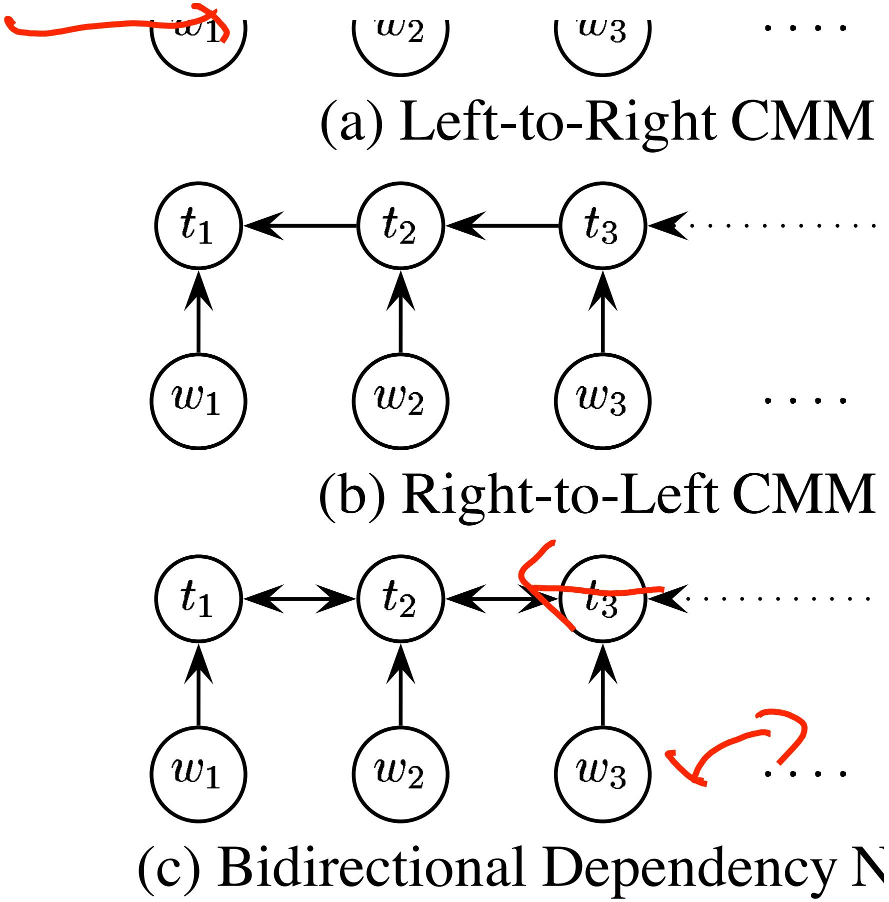


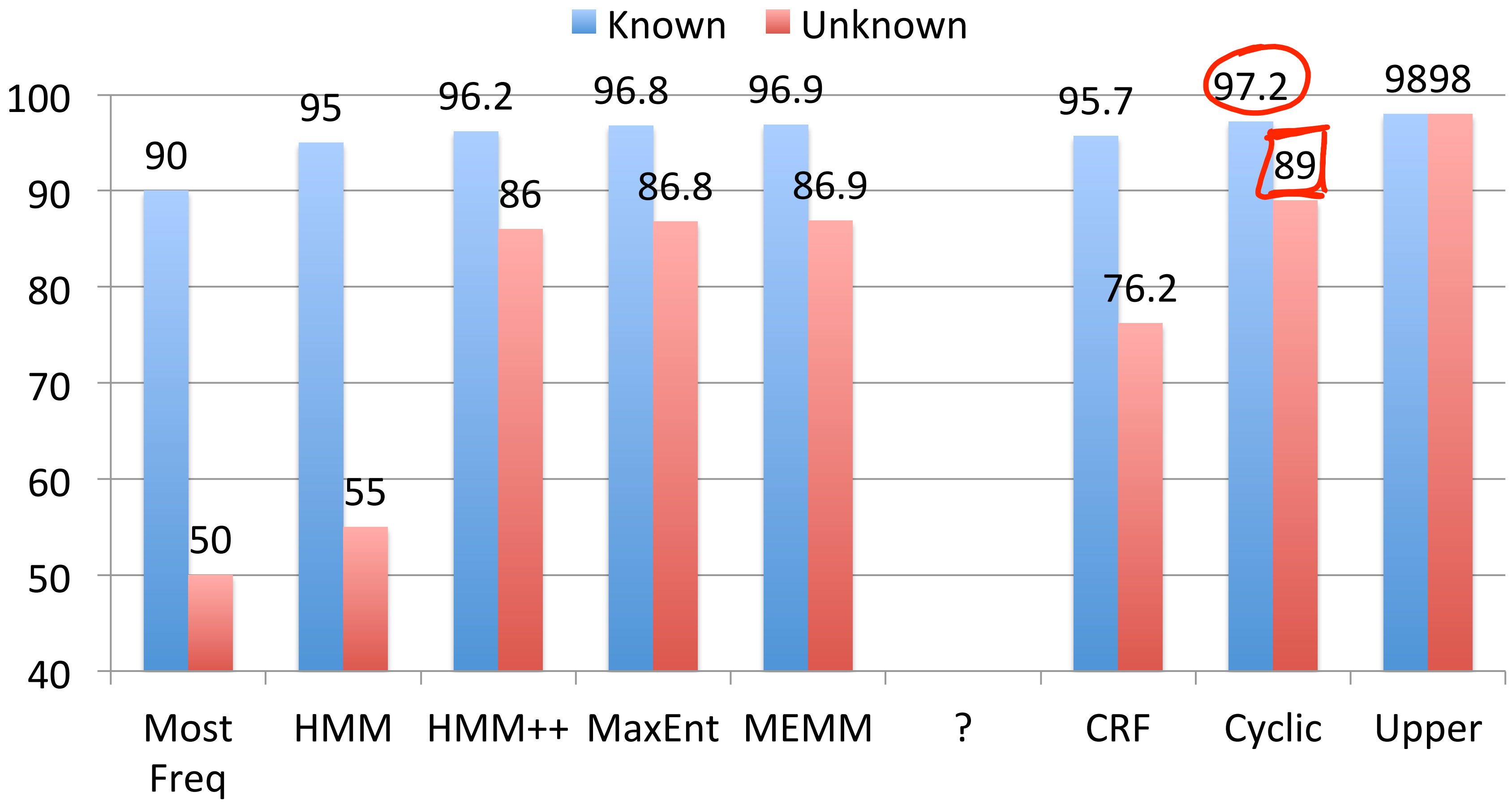
Figure 1: Dependency networks: (a) the (star first-order CMM, (b) the (reversed) right-to-l the bidirectional dependency network.

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:

– Most freq tag:	~90% / ~50%
– Trigram HMM:	~95% / ~55%
– TnT (HMM++):	96.2% / 86.0%
– Maxent $P(s_i\|x)$:	96.8% / 86.8%
– MEMM tagger:	96.9% / 86.9%
– CRF (untuned)	95.7% / 76.2%
– Cyclic tagger:	97.2% / 89.0%
– Upper bound:	~98%

POS Results



Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
96.2
86
96.8
86.8
96.9
86.9
95.7
76.2
97.2
89
9898
Most Freq
HMM
HMM++
MaxEnt
MEMM
?
CRF
Cyclic
Upper

Outline

- POS Tagging
- MaxEnt
- MEMM
- CRFs
- Wrap-up
- Optional: Perceptron

Summary

- Feature-rich models are important!

Outline

- POS Tagging
- MaxEnt
- MEMM
- CRFs
- Wrap-up
- Optional: Perceptron

[Collins 02]

Linear Models: Perceptron

- The perceptron algorithm
  - Iteratively processes the training set, reacting to training errors
  - Can be thought of as trying to drive down training error
- The (online) perceptron algorithm:
  - Start with zero weights
  - Visit training instances (x_i,y_i) one by one
    - Make a prediction

Sentence: $x=x_1 \ldots x_m$

$$y^* = \arg\max_y w \cdot \phi(x_i, y)$$

Tag Sequence: $y=s_1 \ldots s_m$

    - 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?

[Collins 02]

Linear Models: Perceptron

- The perceptron algorithm
  - Iteratively processes the training set, reacting to training errors
  - Can be thought of as trying to drive down training error
- The (online) perceptron algorithm:
  - Start with zero weights $w = 0$
  - Visit training instances (x_i, y_i) one by one
    - Make a prediction

$$y^* = \arg\max_y w \cdot \phi(x_i, y)$$

    - If correct ($y^* == y_i$): no change, goto next example!
    - If wrong: adjust weights

$$w = w + \left(\phi(x_i, y_i) - \phi(x_i, y^*)\right)$$

Challenge: How to compute argmax efficiently?

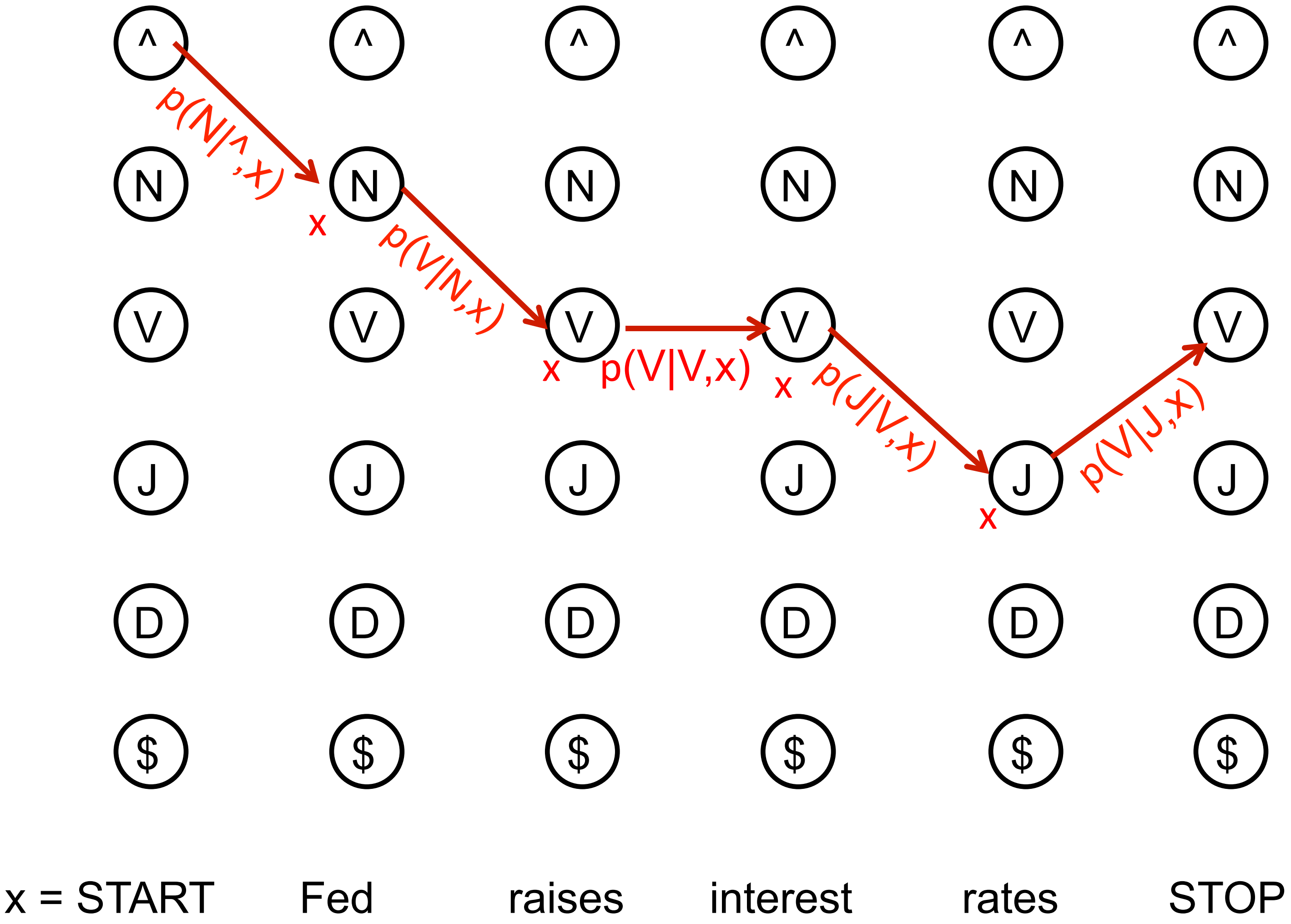
$w + [\alpha \nabla L(w))$

Decoding

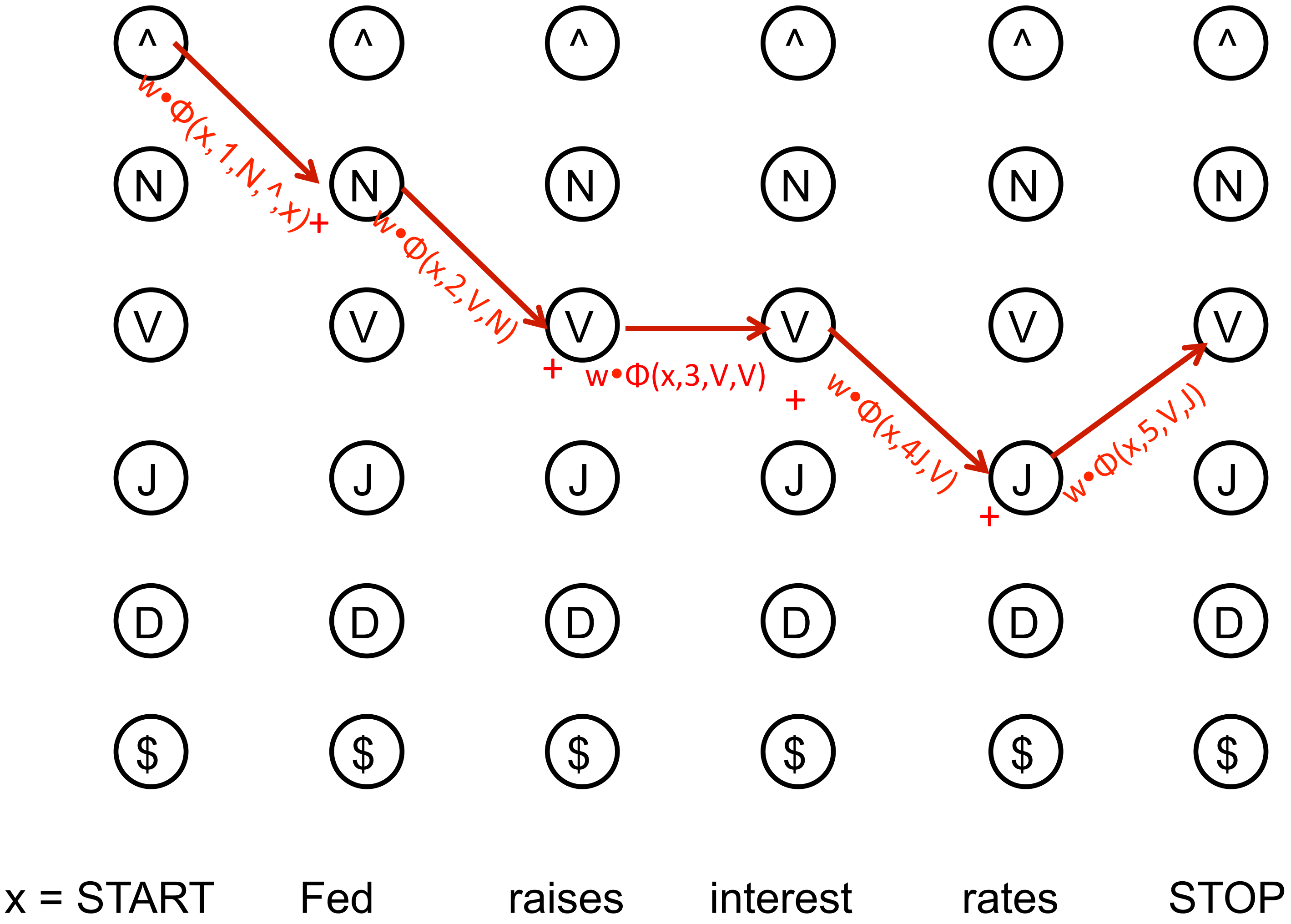
- Linear Perceptron $s^* = \arg\max_s w \cdot \Phi(x, s) \cdot \theta$
  - Features must be local, for x=$x_1 \ldots x_m$, and s=$s_1 \ldots s_m$

$$\underbrace{\Phi(x, s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j)}_{}$$

The MEMM State Lattice / Trellis (repeat)



The Perceptron State Lattice / Trellis



Decoding

- Linear Perceptron $s^* = \arg\max_s w \cdot \Phi(x,s) \cdot \theta$
  - Features must be local, for x=$x_1 \ldots x_m$, and s=$s_1 \ldots s_m$

$$\Phi(x,s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j)$$

  - Define π(i,s_i) to be the max score of a sequence of length i ending in tag s_i

$$\pi(i, s_i) = \max_{s_{i-1}} w \cdot \phi(x, i, s_{i-i}, s_i) + \pi(i-1, s_{i-1})$$

- Viterbi algorithm (HMMs):

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

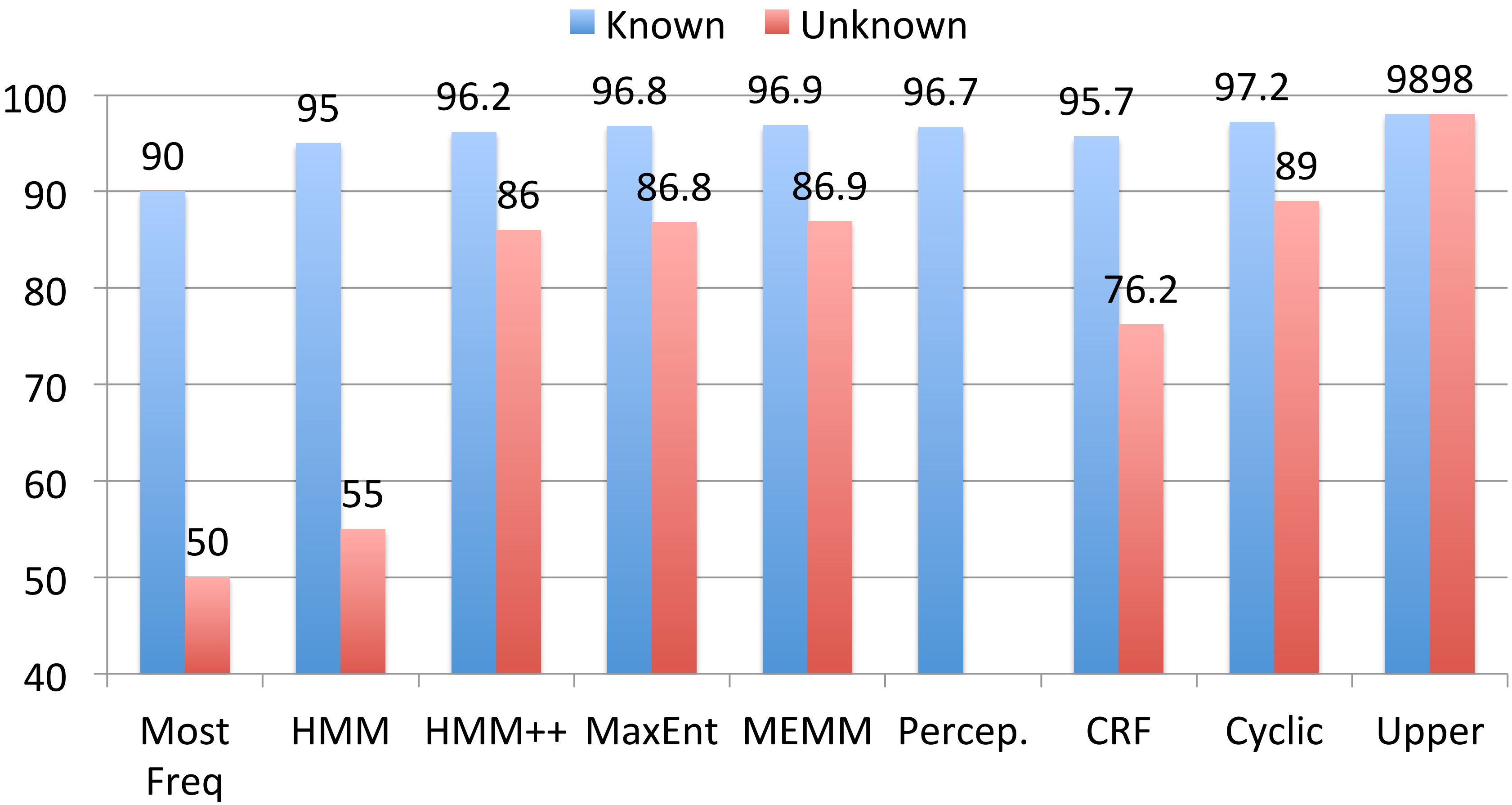
- Viterbi algorithm (Maxent):

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

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent $P(s_i|x)$: 96.8% / 86.8%
  - MEMM tagger: 96.9% / 86.9%
  - Perceptron 96.7% / ??
  - Upper bound: ~98%

POS Results



Known
Unknown
100
90
80
70
60
50
40
90
50
95
55
96.2
86
96.8
86.8
96.9
86.9
96.7
95.7
76.2
97.2
89
9898
Most Freq
HMM
HMM++
MaxEnt
MEMM
Percep.
CRF
Cyclic
Upper