CSP 517
Natural Language Processing
Winter 2015

Parts of Speech

Yejin Choi

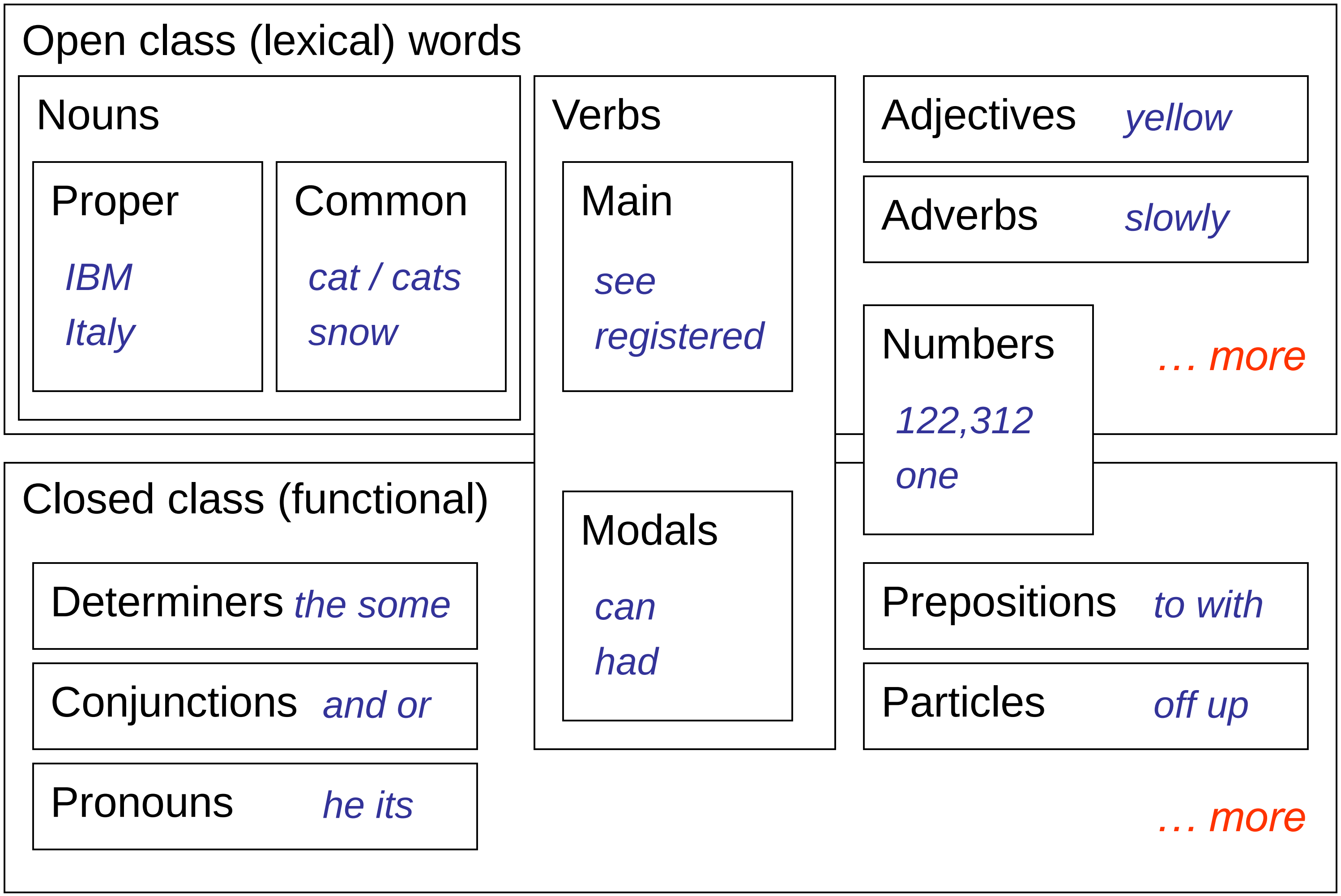
[Slides adapted from Dan Klein, Luke Zettlemoyer]

Overview

- POS Tagging
- Feature Rich Techniques
  - Maximum Entropy Markov Models (MEMMs)
  - Structured Perceptron
  - Conditional Random Fields (CRFs)

Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes



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

Part-of-Speech Ambiguity

- Words can have multiple parts of speech

VBD		VB			
VBN	VBZ	VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN

Fed raises interest rates 0.5 percent

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

- Two basic sources of constraint:
  - Grammatical environment
  - Identity of the current word
- Many more possible features:
  - Suffixes, capitalization, name databases (gazetteers), etc…

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
  - 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 …

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

Ambiguity in POS Tagging

- **Particle (RP) vs. preposition (IN)**
  – He talked *over* the deal.
  – He talked *over* the telephone.
- **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.

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%

Most errors on unknown words

- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text (SOA is ~97.5%)
  - Upper bound: ~98%

Common Errors

- Common errors [from Toutanova & Manning 00]

	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

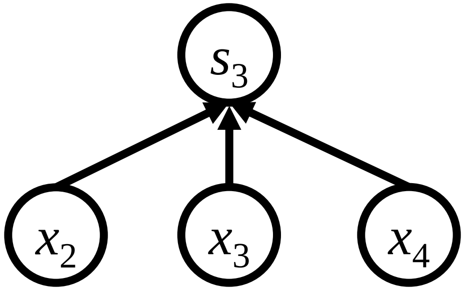
NN/JJ NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

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

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%

  - Q: What does this say about sequence models?
  - Q: How do we add more features to our sequence models?

  - Upper bound: ~98%

MEMM Taggers

- One step up: also condition on previous tags

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

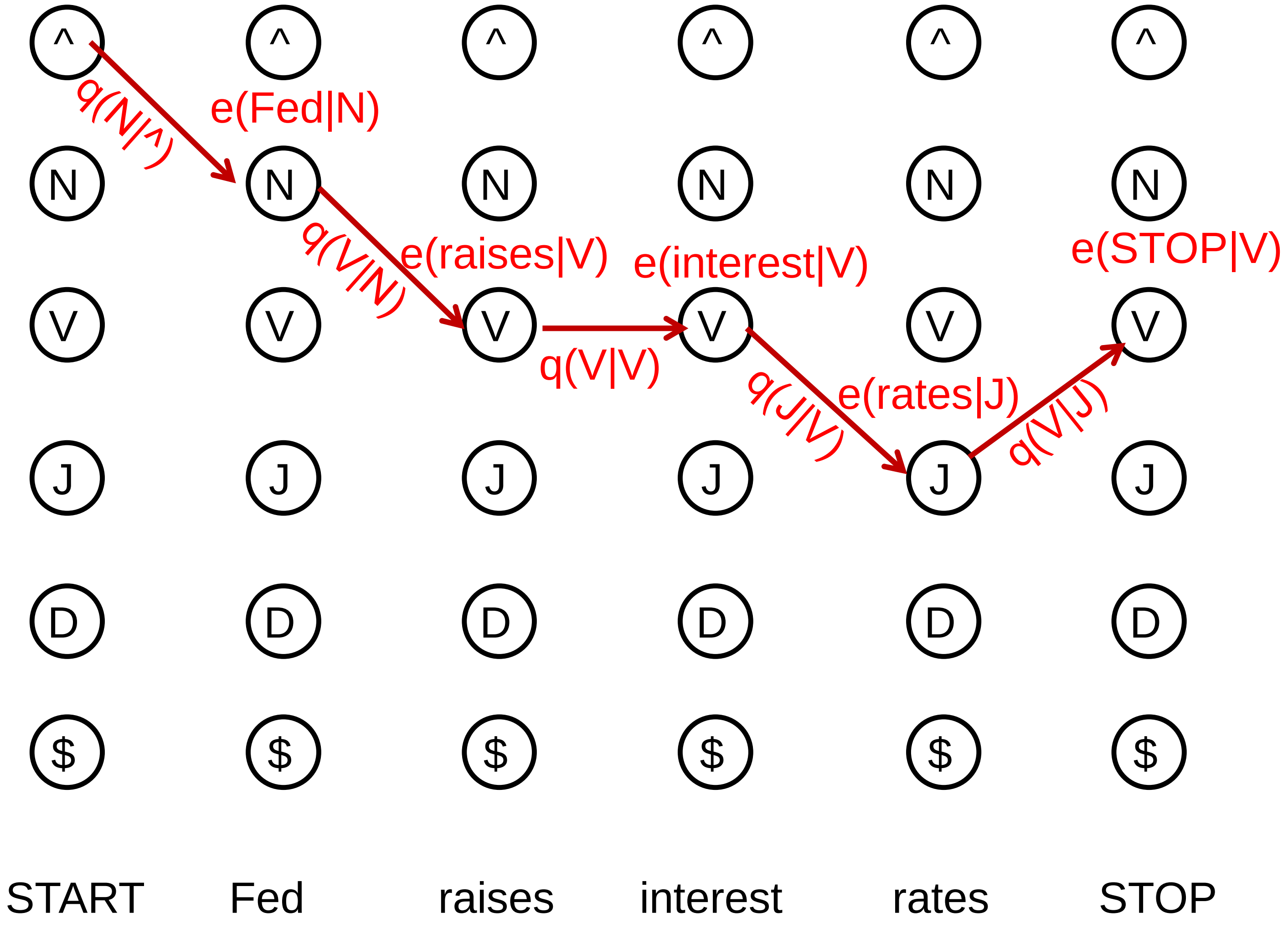
$$= \prod_{i=1}^{m} p(s_i | s_{i-1}, x_1 \dots x_m)$$

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

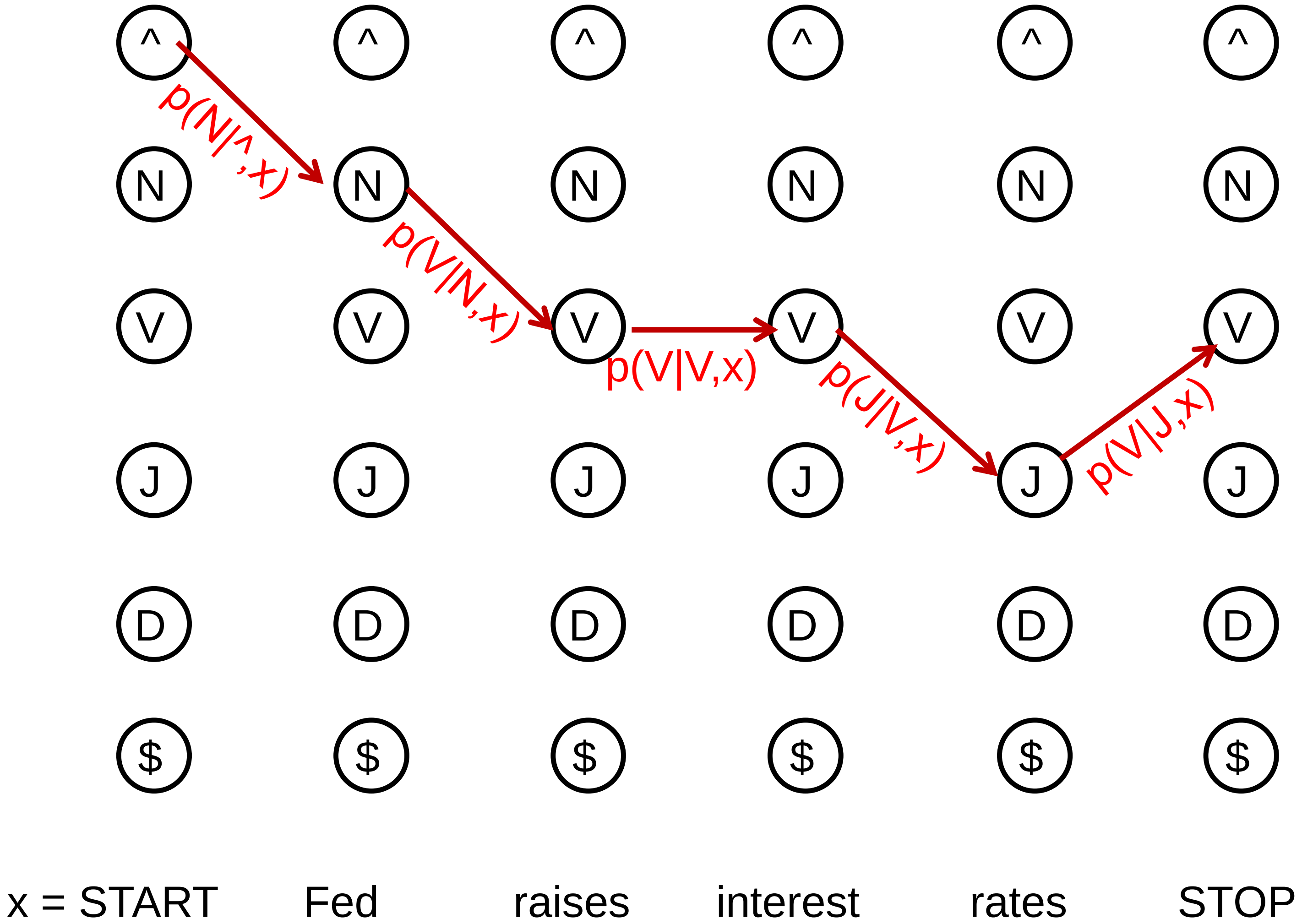
$$p(s_i | s_{i-1}, x_1 \dots x_m) = \frac{\exp\left(w \cdot \phi(x_1 \dots x_m, i, s_{i-1}, s_i)\right)}{\sum_{s'} \exp\left(w \cdot \phi(x_1 \dots x_m, i, s_{i-1}, s')\right)}$$

  - This is referred to as an MEMM tagger [Ratnaparkhi 96]
  - Beam search effective! (Why?)
  - What's the advantage of beam size 1?

The HMM State Lattice / Trellis (repeat slide)



The MEMM State Lattice / Trellis



Decoding

- 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

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

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

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

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%

Global Discriminative Taggers

- Newer, higher-powered discriminative sequence models
  - CRFs (also perceptrons, M3Ns)
  - Do not decompose training into independent local regions
  - Can be deathly slow to train – require repeated inference on training set
- Differences can vary in importance, depending on task
- 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
  - Why isn’t this a big deal for POS tagging?
  - Also: in decoding, condition on predicted, not gold, histories

[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

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

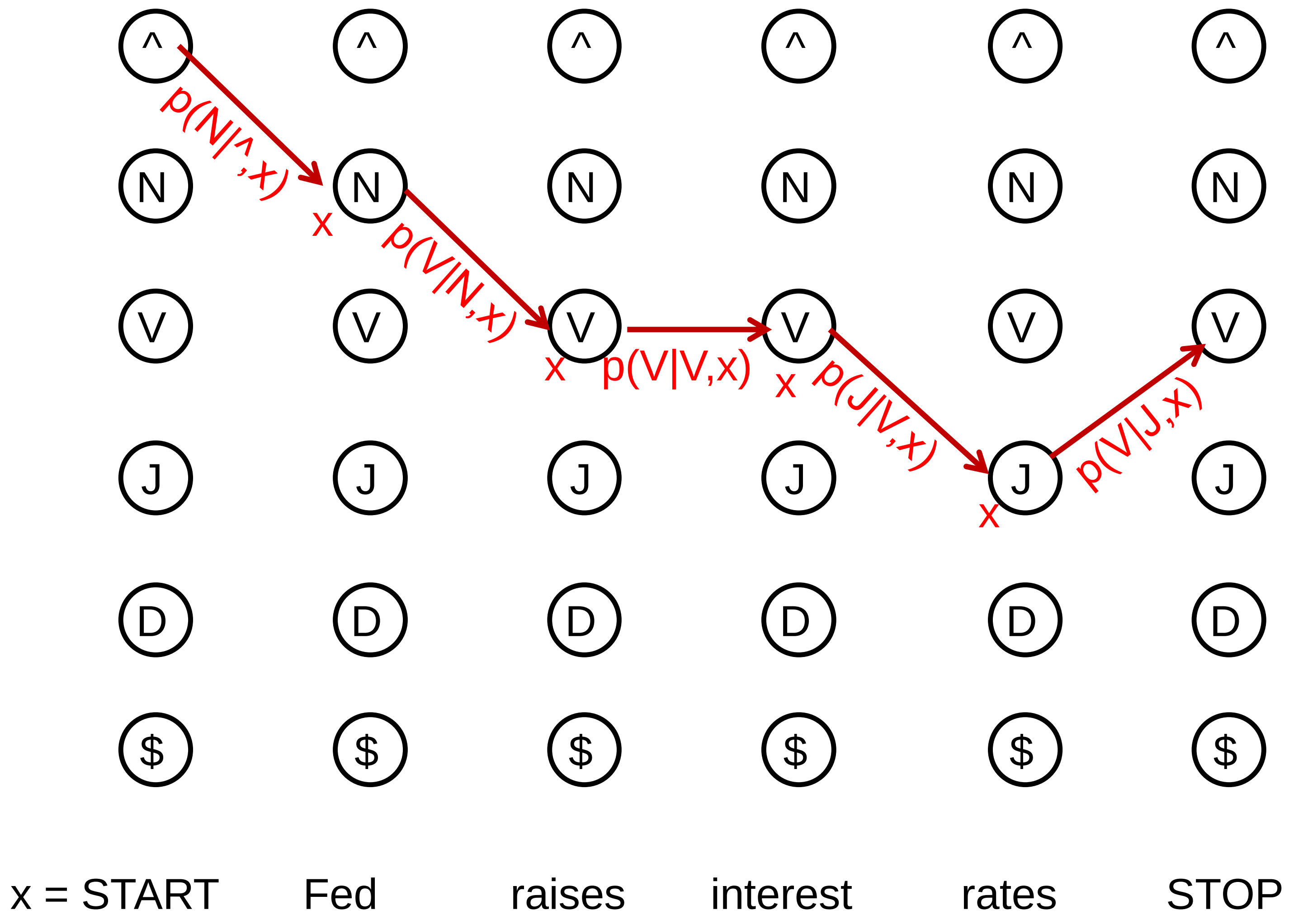
Challenge: How to compute argmax efficiently?

Decoding

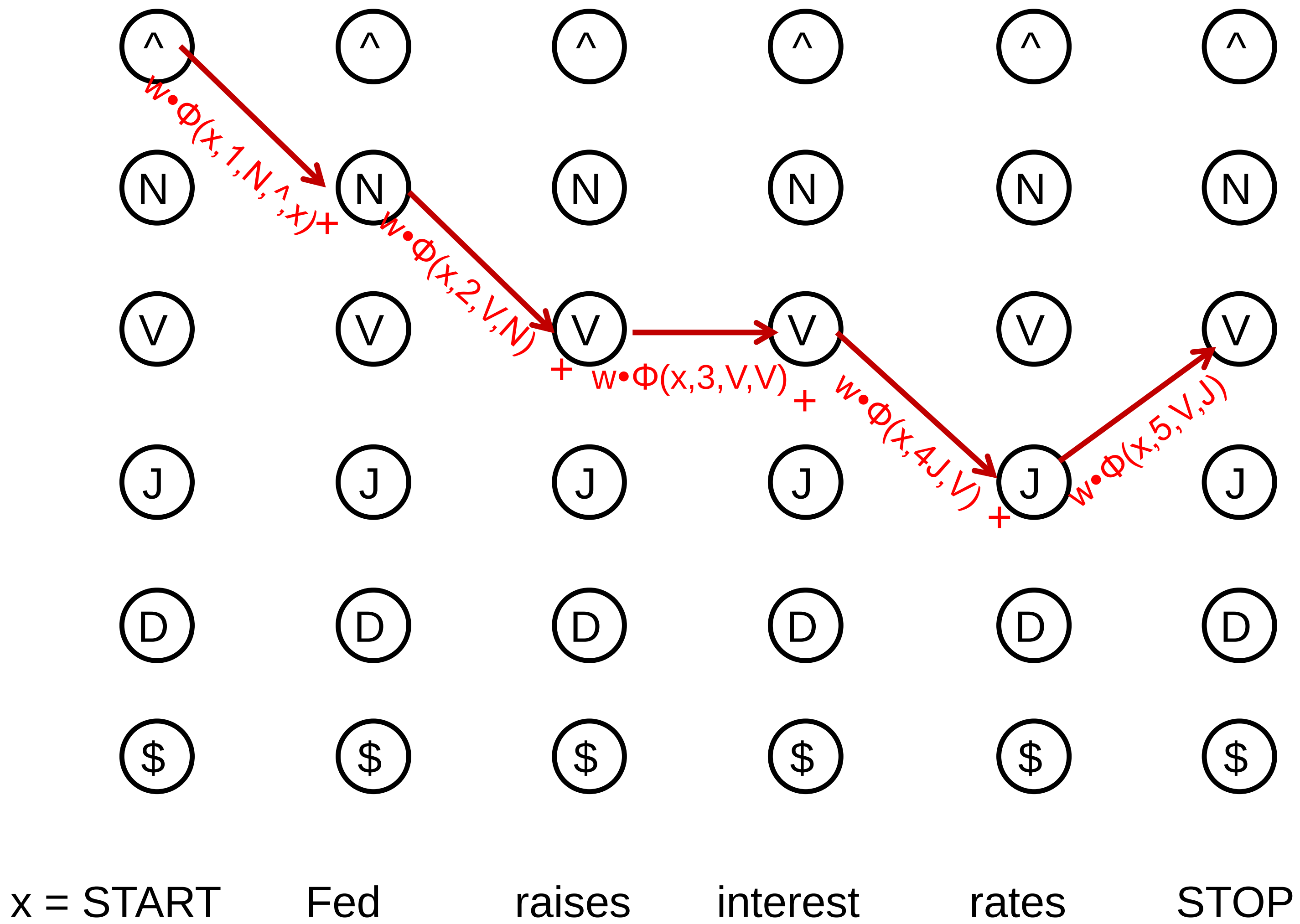
- Linear Perceptron $$s^* = \arg\max_s w \cdot \Phi(x,s) \cdot \theta$$
  - Features must be local, for $x=x_1 \ldots x_m$, and $s=s_1 \ldots s_m$

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

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

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

- 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%

Conditional Random Fields (CRFs)

[Lafferty, McCallum, Pereira 01]

- Maximum entropy (logistic regression)

Sentence: $x=x_1 \dots x_m$

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

Tag Sequence: $y=s_1 \dots s_m$

- 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

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!!!

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

CRFs: Computing Normalization*

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

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

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

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

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

- Forward Algorithm! Remember HMM case:

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

  - Could also use backward?

CRFs: Computing Gradient*

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

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

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

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

- Need forward and backward messages

  See notes for full details!

Overview: Accuracies

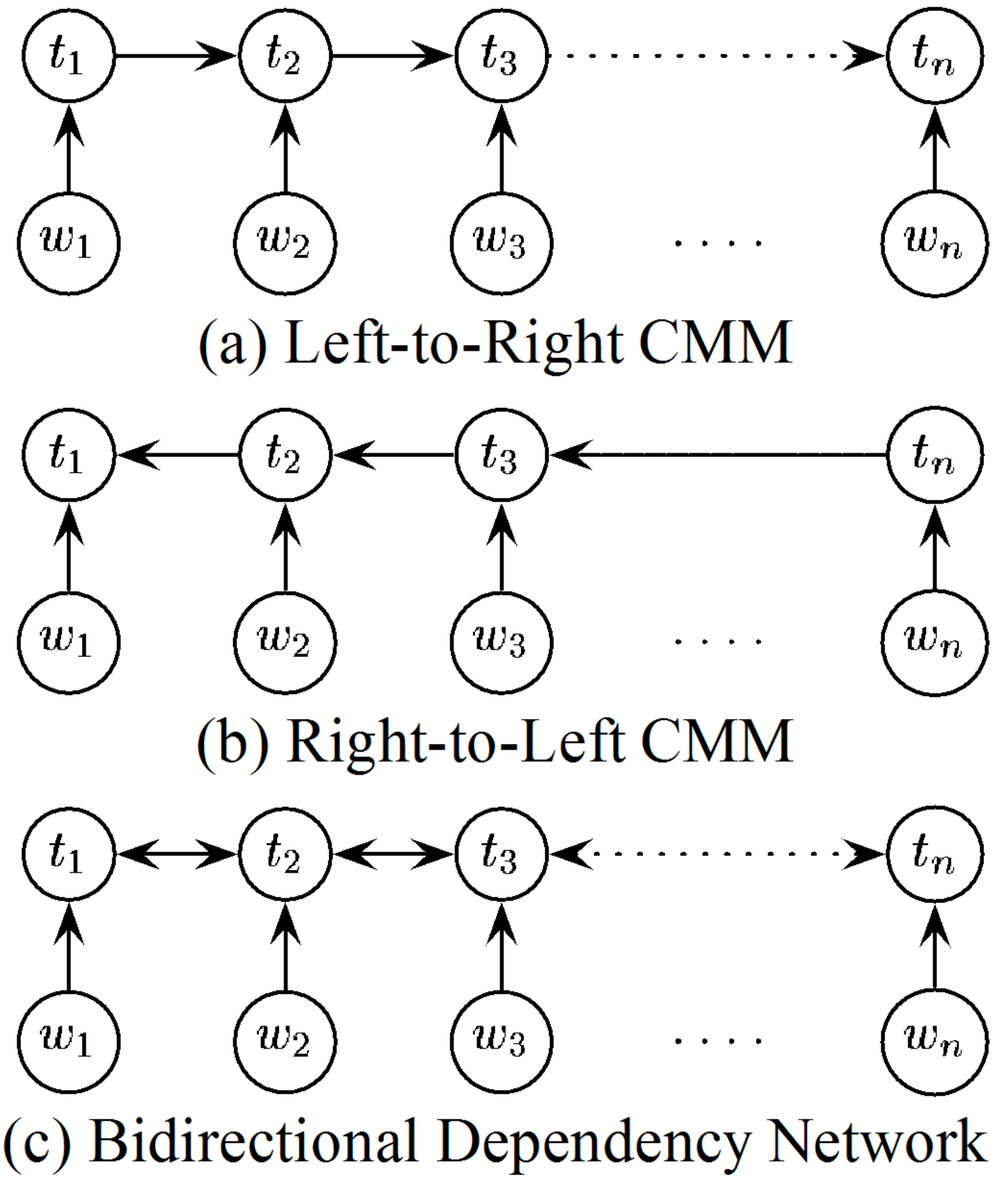
- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent $P(s_i|x)$: 96.8% / 86.8%
  - MEMM tagger: 96.9% / 86.9%
  - Perceptron 96.7% / ??
  - CRF (untuned) 95.7% / 76.2%

  - Upper bound: ~98%

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



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% / ??
  - CRF (untuned) 95.7% / 76.2%
  - Cyclic tagger: 97.2% / 89.0%
  - Upper bound: ~98%

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)