CSEP 517
Natural Language Processing

Linear Sequence Models

Luke Zettlemoyer - University of Washington

[Many slides from Yejin Choi, Dan Klein]

Overview

- Linear Language Model
- Linear Tagging Techniques
  - MEMMs, Structured Perceptron, CRFs
  - Running Example: POS Tagging
- Linear Parsing Model

Review: Language Modeling

- Setup: Assume a (finite) vocabulary of words

$$\mathcal{V} = \{\text{the}, \text{a}, \text{man}, \text{telescope}, \text{Beckham}, \text{two}, \text{Madrid}, \ldots\}$$

- We can construct an (infinite) set of strings

$$\mathcal{V}^{\dagger} = \{\text{the}, \text{a}, \text{the a}, \text{the fan}, \text{the man}, \text{the man with the telescope}, \ldots\}$$

- Data: given a *training set* of example sentences $x \in \mathcal{V}^{\dagger}$
- Problem: estimate a probability distribution

$$\sum_{x \in \mathcal{V}^{\dagger}} p(x) = 1$$

and $p(x) \geq 0$ for all $x \in \mathcal{V}^{\dagger}$

$p(\text{the}) = 10^{-12}$

$p(\text{a}) = 10^{-13}$

$p(\text{the fan}) = 10^{-12}$

$p(\text{the fan saw Beckham}) = 2 \times 10^{-8}$

$p(\text{the fan saw saw}) = 10^{-15}$

$\ldots$

- Question: can we do better than n-grams, now that we have features?

Log-linear LMs

- Law of conditional probability:

$$p(x_1 \dots x_n) = \prod_{i=1}^{n} q(x_i | x_1 \dots x_{i-1})$$

  - Approach: train $q(x_i|x_1 \dots x_{i-1})$ as a discrete log-linear (maxent) model:

$$q(x_i | x_1 \dots x_{i-1}) = \frac{\exp(w \cdot \phi(x_1 \dots x_i))}{\sum_{x'} \exp(w \cdot \phi(x_1 \dots x_{i-1}, x')))}$$

  - Can train with any techniques from before, e.g. maxent, perceptron, etc.
  - Was SOTA before NN methods [e.g. Roark et al 2006]

Example: Trigger-based Linear Models

[Rosenfeld, 1996]

- Features: carefully chosen word pairs (called triggers) that appear anywhere before target word in sentence (or document)

HARVEST ⇐ CROP HARVEST CORN SOYBEAN SOYBEANS AGRICULTURE GRAIN DROUGHT GRAINS BUSHELS

HAVANA ⇐ CUBAN CUBA CASTRO HAVANA FIDEL CASTRO'S CUBA'S CUBANS COMMUNIST MIAMI REVOLUTION

- Results:

vocabulary	top 20,000 words of WSJ corpus	
training set	5MW (WSJ)	
test set	325KW (WSJ)	
trigram perplexity (baseline)	173	173
ME experiment	top 3	top 6
ME constraints:		
unigrams	18400	18400
bigrams	240000	240000
trigrams	414000	414000
triggers	36000	65000
ME perplexity	134	130
perplexity reduction	23%	25%
0.75·ME + 0.25·trigram perplexity	129	127
perplexity reduction	25%	27%

Table 8: Maximum Entropy models incorporating N-gram and trigger constraints.

Review: Pairs of Sequences

- 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 …

- We previously learn a joint distribution:

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

  - And then computed the most likely assignment:

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

- Q: Can we do better, now that we have feature rich models?

Why POS Tagging?

- Useful in and of itself (more than you'd think)
  - Text-to-speech: record, lead
  - Lemmatization: saw[v] $\rightarrow$ see, saw[n] $\rightarrow$ 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

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

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

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

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

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

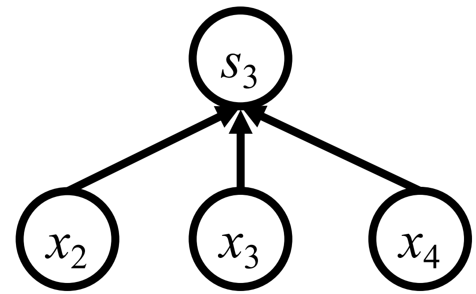
NN/JJ NN
official knowledge

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

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

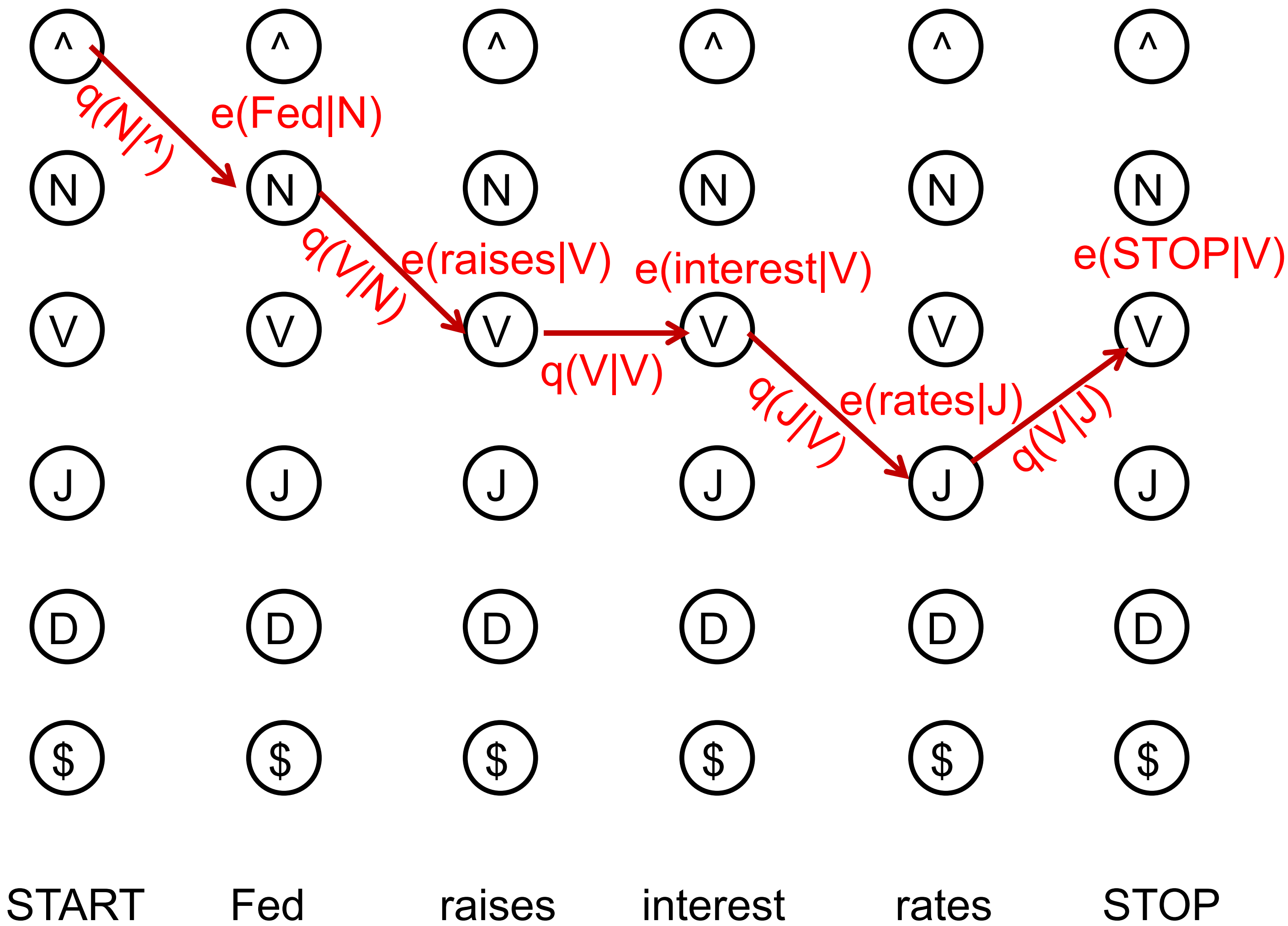
$$= \prod_{i=1}^{m} p(s_i | s_{i-1}, x_1 \ldots x_m)$$

  - Train up $p(s_i|s_{i-1},x_1..x_m)$ as a discrete log-linear (maxent) model, then use to score sequences

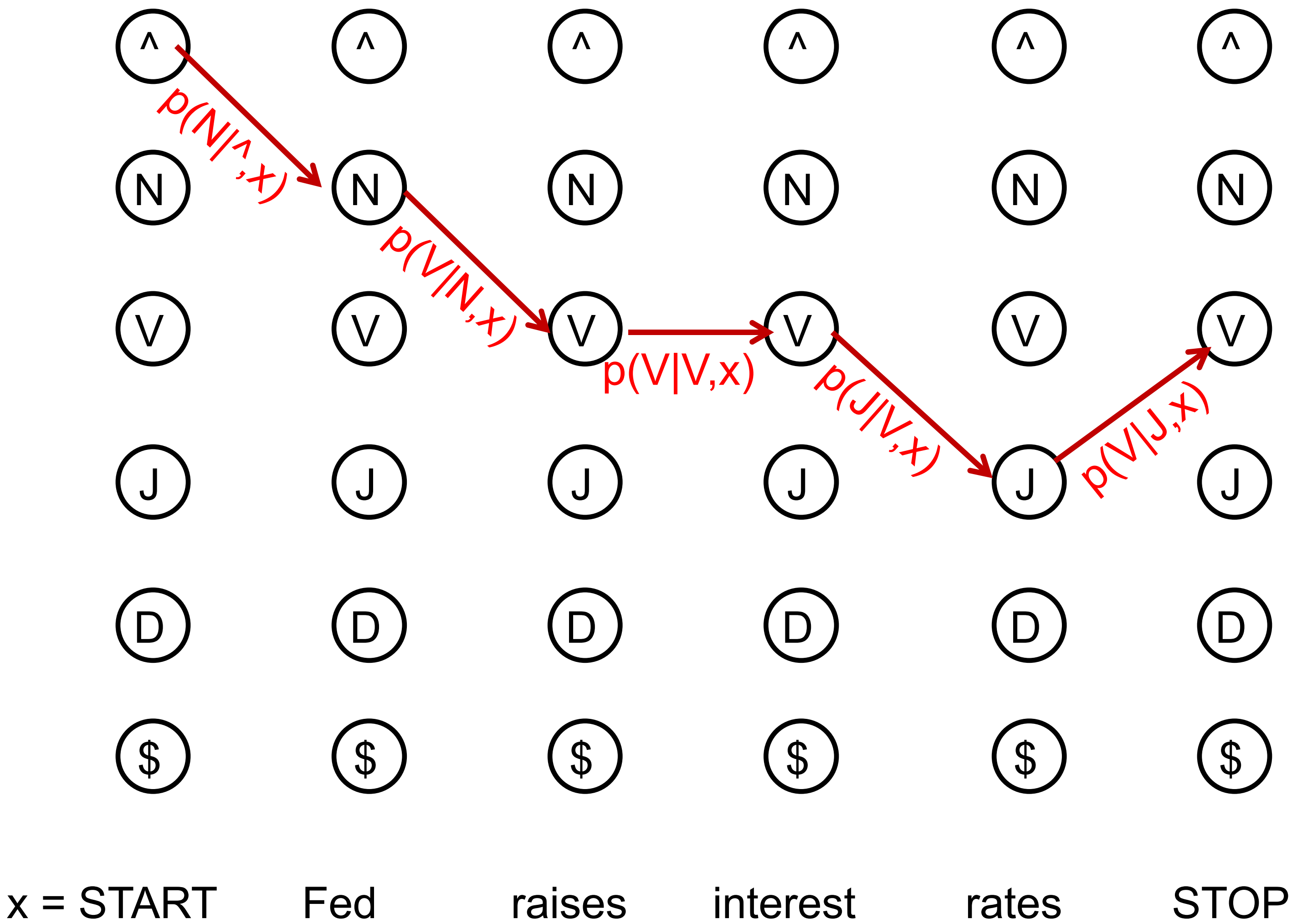
$$p(s_i | s_{i-1}, x_1 \ldots x_m) = \frac{\exp\left(w \cdot \phi(x_1 \ldots x_m, i, s_{i-1}, s_i)\right)}{\sum_{s'} \exp\left(w \cdot \phi(x_1 \ldots x_m, i, s_{i-1}, s')\right)}$$

  - 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



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

Review: Discrete 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

Previously assumed y comes from a small set.

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

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

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

- Question: What if y is a sequence instead?

Structured Perceptron

[Collins 02]

- 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: $x = x_1 \dots x_m$

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

Tag Sequence: $y = s_1 \dots s_m$

    - 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?

Local Features

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

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

  - Will be important for efficient inference, but lets look at some examples first

HMM Recap: Chunking

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

Germany/BL ’s/NA representative/NA to/NA the/NA European/BO Union/CO ’s/NA veterinary/NA committee/NA Werner/BP Zwingman/CP said/NA on/NA Wednesday/NA consumers/NA should/NA…

- 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

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

- Can we mimic the parameters of HMM?
  - Transitions: $\phi_{t1,t2}(s_{j-1},s_j)=1$ if s_{j-1}==t1 AND s_j==t1, else 0
    - instantiate for all pairs of tags t1, t2
    - e.g. $\phi_{CO,BO}(s_{j-1},s_j)=1$ if s_{j-1}==CO AND s_j==BO, else 0
  - Emissions: $\phi_{t,w}(x_j,s_j)=1$ if s_j==t AND x_j==w, else 0
    - instantiate for all pairs of word w and tag t
    - e.g. $\phi_{BL,Seattle}(x_j,s_j)=1$ if s_j==BL AND x_j==Seattle, else 0
- Can also have lots of other features, for example:
  - $\phi_7(s_j)=1$ if j==3 AND s_j ==NA, else 0 [this is not a good feature, but allowed!]
  - $\phi_{cap}(x_j,s_j)=1$ if x_j is capitalized AND (s_j == *L OR s_j == *P), else 0 [probably a good feature, if you know those classes tend to be capitalized. shared parameter across many words and tags]

Review: Discrete Log-linear Models

- Maximum entropy (logistic regression)
  - Model: use the scores as probabilities:

Previously assumed y comes from a small set.

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

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

$$L(w) = \sum_{i=1}^{n} \log p(y_i|x_i;w) \qquad w^* = \arg\max_w L(w)$$

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

  - Prediction: output argmax_y p(y|x;w)
  - Question: What if y is a sequence instead?

Conditional Random Fields (CRFs)

[Lafferty, McCallum, Pereira 01]

- Maximum entropy (logistic regression)

Sentence: $x=x_1 \ldots x_m$

Tag Sequence: $y=s_1 \ldots s_m$

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

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

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

- Prediction: output argmax_y p(y|x;w)

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

Review PCFGs

- Model
  - The probability of a tree t with n rules $\alpha_i \rightarrow \beta_i$, i = 1..n

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

- Learning
  - Read the rules off of labeled sentences, use ML estimates for probabilities

$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

  - and use all of our standard smoothing tricks!

- Inference
  - For input sentence s, define T(s) to be the set of trees whole *yield* is s (whole leaves, read left to right, match the words in s)

$$t^*(s) = \arg\max_{t \in \mathcal{T}(s)} p(t)$$

Review: PCFG Example

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

- Probability of a tree t with rules

$$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$$

is

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$.

Linear CFG

- Key Assumption
  - Features for a tree t with n rules $\alpha_i \rightarrow \beta_i$, i = 1..n

$$\Phi(t, s) = \sum_{i=1}^{n} \phi(\alpha_i \rightarrow \beta_i, s)$$

- Model and Learning
  - Can define log=linear model, perceptron score, etc.

$$score(t, s) = w \cdot \Phi(t, s) \qquad p(t|s) = \frac{\exp(w \cdot \Phi(t, s))}{\sum_{t' \in \mathcal{T}(s)} \exp(w \cdot \Phi(t', s))}$$

- Inference
  - Can adapt CKY and Inside-Outside algorithms, as long as feature assumption (above) is true

Final Results

Parser	F1 ≤ 40 words	F1 all words
Klein & Manning '03	86.3	85.7
Matsuzaki et al. '05	86.7	86.1
Collins '99	88.6	88.2
Charniak & Johnson '05	90.1	89.6
Petrov et. al. 06	90.2	89.7
Finkel et. al. 08	89	88

End of Deck: Everything else is optinal

Decoding

- Linear Perceptron $$s^* = \arg\max_s w \cdot \Phi(x, s)$$
  - 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

Review: HMMs

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

$$y* = \underset{y_1 \ldots y_n}{\operatorname{argmax}}\ p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

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

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

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

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

- We now have an efficient algorithm. Start with i=0 and work your way to the end of the sentence!

Dynamic Program: Structured Perceptron

$$w \cdot \Phi(x_1 \dots x_n, s_1 \dots s_n) = \sum_{j=1}^{n} w \cdot \phi(x, j, s_{j-1}, s_j)$$

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

$$\pi(i, s_i) = \max_{s_1 \dots s_{i-1}} w \cdot \Phi(x_1 \dots x_i, s_1 \dots s_i)$$

$$= \max_{s_{i-1}} w \cdot \phi(x, i, s_{i-1}, s_i) + \max_{s_1 \dots s_{i-2}} w \cdot \Phi(x_1 \dots x_{i-1}, s_1 \dots s_{i-1})$$

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

- We now have an efficient algorithm. Start with i=0 and work your way to the end of the sentence!

Decoding

- Linear Perceptron $s^* = \arg\max_s w \cdot \Phi(x, s)$
  - 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

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

- 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%

Review: Discrete Log-linear Models

- Maximum entropy (logistic regression)
  - Model: use the scores as probabilities:

Previously assumed y comes from a small set.

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

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

$$L(w) = \sum_{i=1}^{n} \log p(y_i|x_i;w) \qquad w^* = \arg\max_w L(w)$$

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

  - Prediction: output $\text{argmax}_y\ p(y|x;w)$
  - Question: What if y is a sequence instead?

Conditional Random Fields (CRFs)

[Lafferty, McCallum, Pereira 01]

- 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 $\{(x_i, y_i)\}_{i=1}^n$

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

  - Prediction: output argmax_y p(y|x;w)

- 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 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

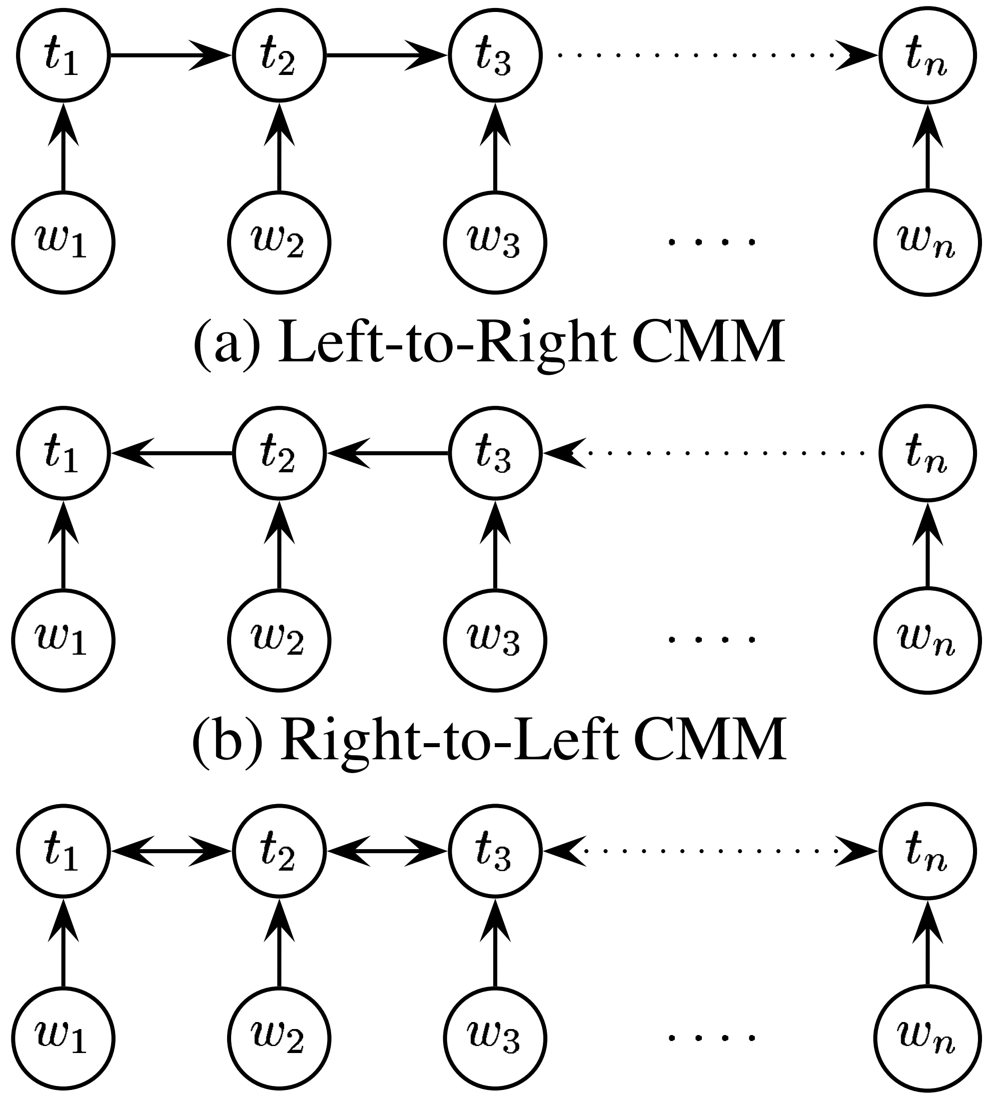
- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent $P(s_i|x)$: 96.8% / 86.8%
  - MEMM tagger: 96.9% / 86.9%
  - Perceptron 96.7% / ??
  - CRF (untuned) 95.7% / 76.2%

  - 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



(a) Left-to-Right CMM

(b) Right-to-Left CMM

(c) 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%
  - 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)

Review PCFGs

- Model
  - The probability of a tree t with n rules $\alpha_i \rightarrow \beta_i$, i = 1..n

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

- Learning
  - Read the rules off of labeled sentences, use ML estimates for probabilities

$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

  - and use all of our standard smoothing tricks!

- Inference
  - For input sentence s, define T(s) to be the set of trees whole *yield* is s (whole leaves, read left to right, match the words in s)

$$t^*(s) = \arg\max_{t \in \mathcal{T}(s)} p(t)$$

Review: PCFG Example

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

- Probability of a tree t with rules

$$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$$

is

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$.

Linear CFG

- Key Assumption
  - Features for a tree t with n rules $\alpha_i \rightarrow \beta_i$, i = 1..n

$$\Phi(t,s) = \sum_{i=1}^{n} \phi(\alpha_i \rightarrow \beta_i, s)$$

- Model and Learning
  - Can define log=linear model, perceptron score, etc.

$$score(t,s) = w \cdot \Phi(t,s) \qquad p(t|s) = \frac{\exp(w \cdot \Phi(t,s))}{\sum_{t' \in \mathcal{T}(s)} \exp(w \cdot \Phi(t',s))}$$

- Inference
  - Can adapt CKY and Inside-Outside algorithms, as long as feature assumption (above) is true

Log-Linear CFG [Finkel et al 2008]

- Features

Table 1: Lexicon and grammar features. w is the word and t the tag. r represents a particular rule along with span/split information; ρ is the rule itself, r_p is the parent of the rule; w_b, w_s, and w_e are the first, first after the split (for binary rules) and last word that a rule spans in a particular context. All states, including the POS tags, are annotated with parent information; $b(s)$ represents the base label for a state s and $p(s)$ represents the parent annotation on state s. $ds(w)$ represents the distributional similarity cluster, and $lc(w)$ the lower cased version of the word, and $unk(w)$ the unknown word class.

Lexicon Features	Grammar Features	
t		Binary-specific features
$b(t)$	ρ	
$\langle t,w\rangle$	$\langle b(p(r_p)),ds(w_s)\rangle$	$\langle b(p(r_p)),ds(w_{s-1},dsw_s)\rangle$
$\langle t,lc(w)\rangle$	$\langle b(p(r_p)),ds(w_e)\rangle$	**PP feature:**
$\langle b(t),w\rangle$	**unary?**	if right child is a PP then $\langle r,w_s\rangle$
$\langle b(t),lc(w)\rangle$	**simplified rule:**	**VP features:**
$\langle t,ds(w)\rangle$	base labels of states	if some child is a verb tag, then rule,
$\langle t,ds(w_{-1})\rangle$	**dist sim bigrams:**	with that child replaced by the word
$\langle t,ds(w_{+1})\rangle$	all dist. sim. bigrams below	
$\langle b(t),ds(w)\rangle$	rule, and base parent state	Unaries which span one word:
$\langle b(t),ds(w_{-1})\rangle$	**dist sim bigrams:**	
$\langle b(t),ds(w_{+1})\rangle$	same as above, but trigrams	$\langle r,w\rangle$
$\langle p(t),w\rangle$	**heavy feature:**	$\langle r,ds(w)\rangle$
$\langle t,unk(w)\rangle$	whether the constituent is "big"	$\langle b(p(r)),w\rangle$
$\langle b(t),unk(w)\rangle$	as described in (Johnson, 2001)	$\langle b(p(r)),ds(w)\rangle$

Final Results

Parser	F1 ≤ 40 words	F1 all words
Klein & Manning ’03	86.3	85.7
Matsuzaki et al. ’05	86.7	86.1
Collins ’99	88.6	88.2
Charniak & Johnson ’05	90.1	89.6
Petrov et. al. 06	90.2	89.7
Finkel et. al. 08	89	88