Algorithms for NLP

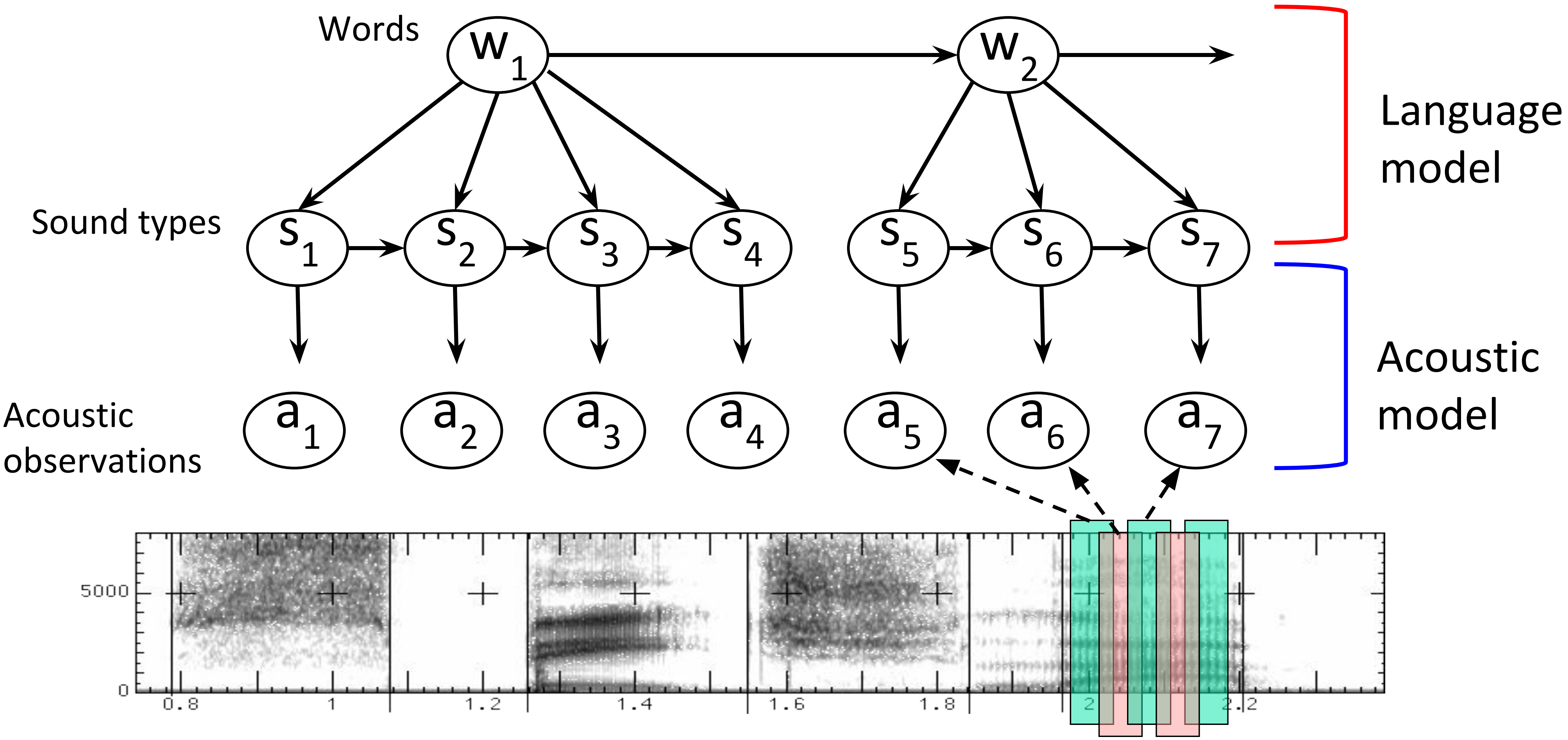
Part of Speech, NER, CRF

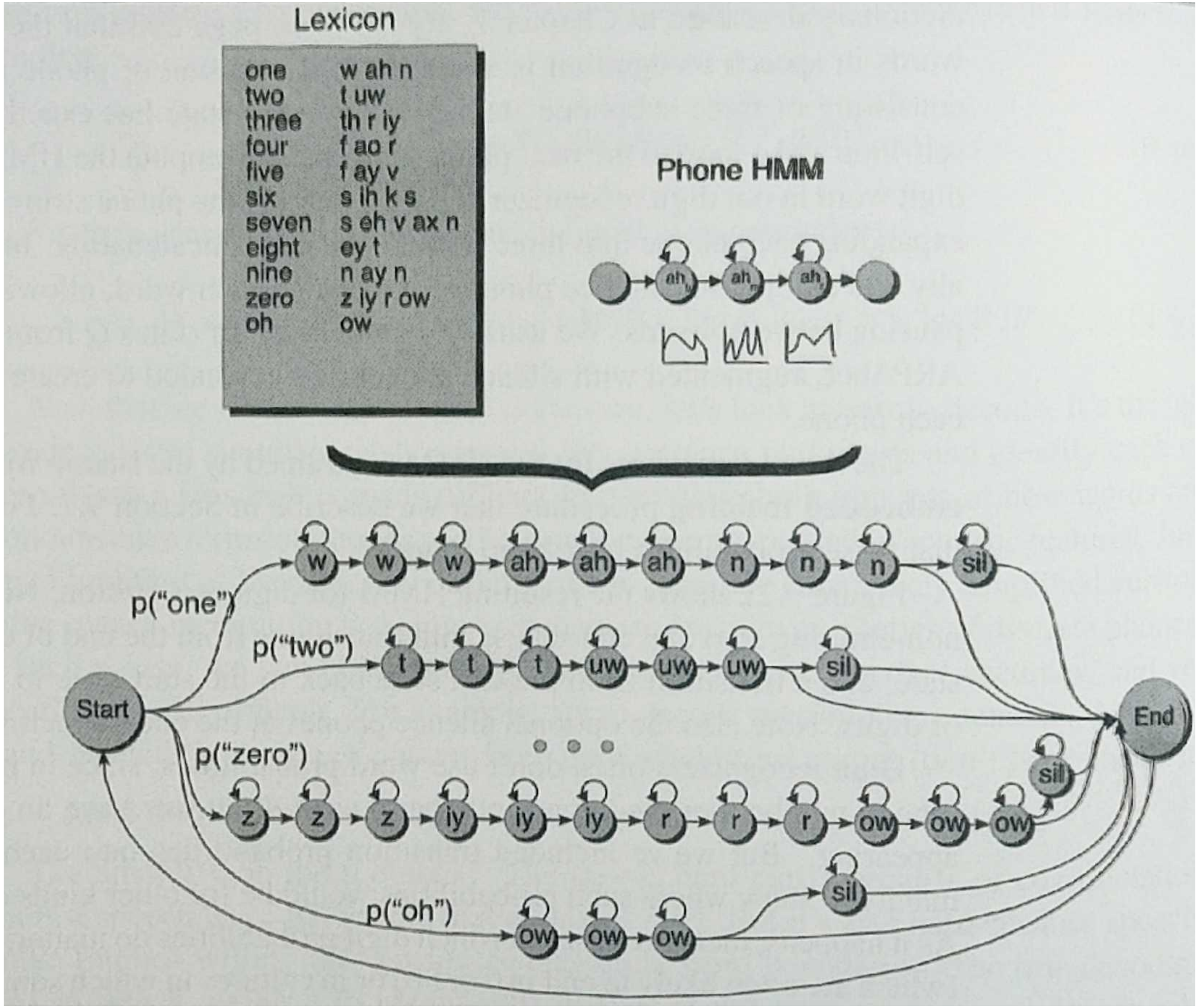
Aldrian Obaja Muis – CMU

Slides adapted from: Dan Klein – UC Berkeley

Taylor Berg-Kirkpatrick, Yulia Tsvetkov – CMU

Speech Model





Lexicon
one w ah n
two t uw
three th r iy
four f ao r
five f ay v
six s ih k s
seven s eh v ax n
eight ey t
nine n ay n
zero z iy r ow
oh ow
Phone HMM
Start
End
p("one")
p("two")
p("zero")
p("oh")
w w w ah ah ah n n n sil
t t t uw uw uw sil
z z z iy iy iy r r r ow ow ow sil
ow ow ow sil

Parts of Speech

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*
  - Auxiliary: *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

English Penn Treebank Part-of-Speech

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*'s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Part-of-Speech in Many 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

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
  - Text-to-speech: how to pronounce “record”, “lead”, “read”?
  - 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

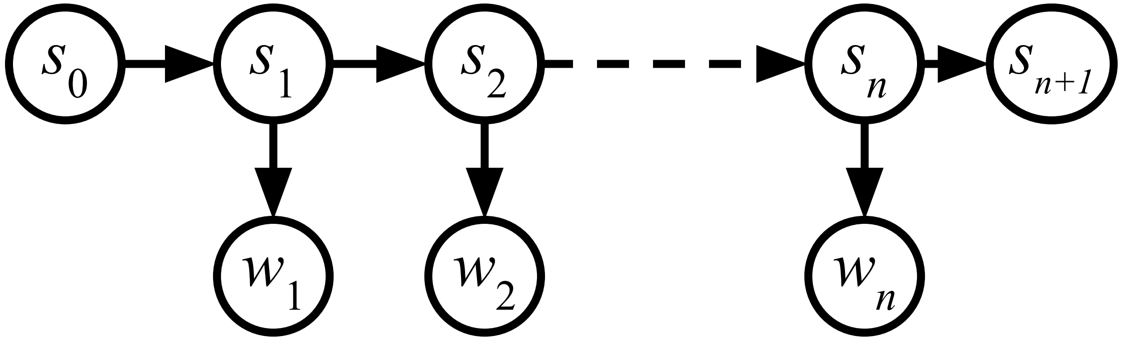
```
                                   IN
DT   NNP     NN   VBD VBN  RP  NN       NNS
The Georgia branch had taken on loan commitments …
```

```
          VBN
DT   NN   VBD  IN  DT  NN VBD
The horse raced past the barn fell
```

Part-of-Speech Tagging

Classic Solution: HMMs

- We want a model of sequences s and observations w

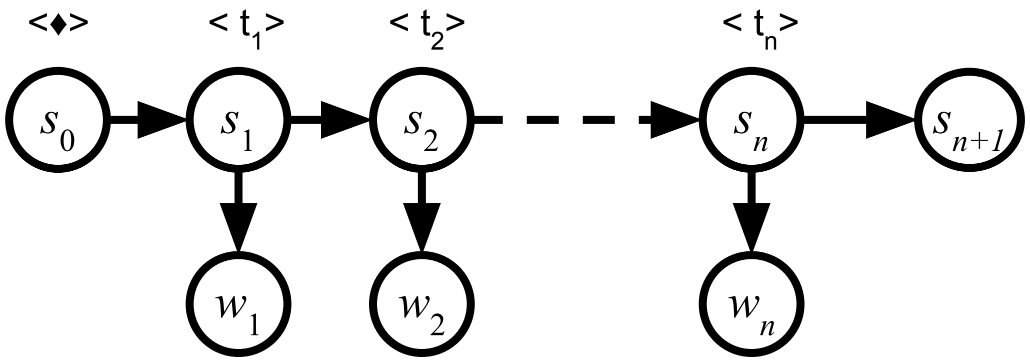


$$P(\mathbf{s}, \mathbf{w}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)$$

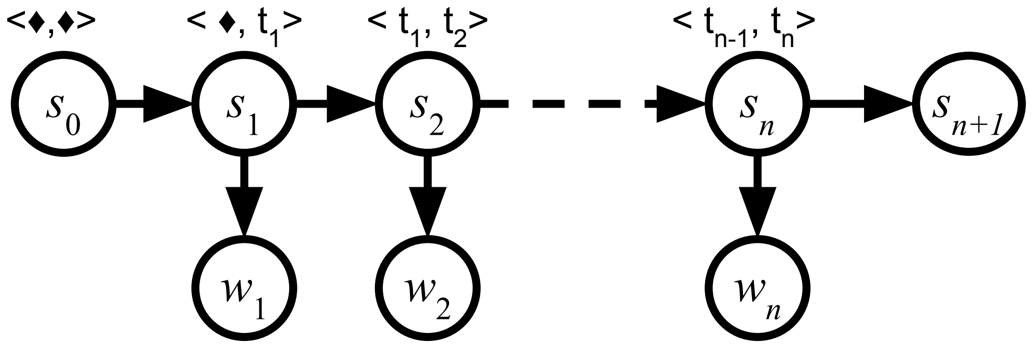
- Assumptions:
  - States are tag n-grams
  - Usually add a dedicated start (s_0) and end state (s_{n+1})
  - Tag/state sequence is generated by a Markov model
  - Words are chosen independently, conditioned only on the tag/state
  - These are totally broken assumptions: why?

States

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



  - In a trigram tagger, states = tag pairs



Estimating Transitions

- Use standard smoothing methods to estimate transitions:

$$P(t_i \mid t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i \mid t_{i-1}, t_{i-2}) + \lambda_1 \hat{P}(t_i \mid t_{i-1}) + (1 - \lambda_1 - \lambda_2)\hat{P}(t_i)$$

- Can get a lot fancier (e.g. KN smoothing) or use higher orders, but in this case it doesn't buy much

- One option: encode more into the state, e.g. whether the previous word was capitalized (Brants 00) – State Splitting

- BIG IDEA: The basic approach of state-splitting / refinement turns out to be very important in a range of tasks (e.g., the states we saw in speech)

Estimating Emissions

$$P(\mathbf{s}, \mathbf{w}) = \prod_i P(s_i | s_{i-1}) P(w_i | s_i)$$

- Emissions are trickier:
  - Words we've never seen before
  - Words which occur with tags we've never seen them with
  - One option: break out the fancy smoothing (e.g. KN, Good-Turing)
  - Issue: unknown words aren't black boxes:

    343,127.23 11-year Minteria reintroducibly

  - Basic solution: unknown words classes (affixes or shapes)

    $D^+,D^+.D^+$ $D^+\text{-}x^+$ Xx^+ x^+-"ly"

  - Common approach: Estimate P(t|w) and invert
  - [Brants 00] used a suffix trie as its (inverted) emission model

Disambiguation (Inference)

- Problem: find the most likely sequence under the model

$$t^* = \arg\max_{\mathbf{t}} P(\mathbf{t}|\mathbf{w})$$

- Given model parameters, we can score any tag sequence: P(t|w) -> likelihood

<♦,♦>	<♦,NNP>	<NNP, VBZ>	<VBZ, NN>	<NN, NNS>	<NNS, CD>	<CD, NN>	<STOP>
	NNP	VBZ	NN	NNS	CD	NN	.
	Fed	raises	interest	rates	0.5	percent	.

P(NNP|<♦,♦>) P(Fed|NNP) P(VBZ|<NNP,♦>) P(raises|VBZ) P(NN|VBZ,NNP)…..

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

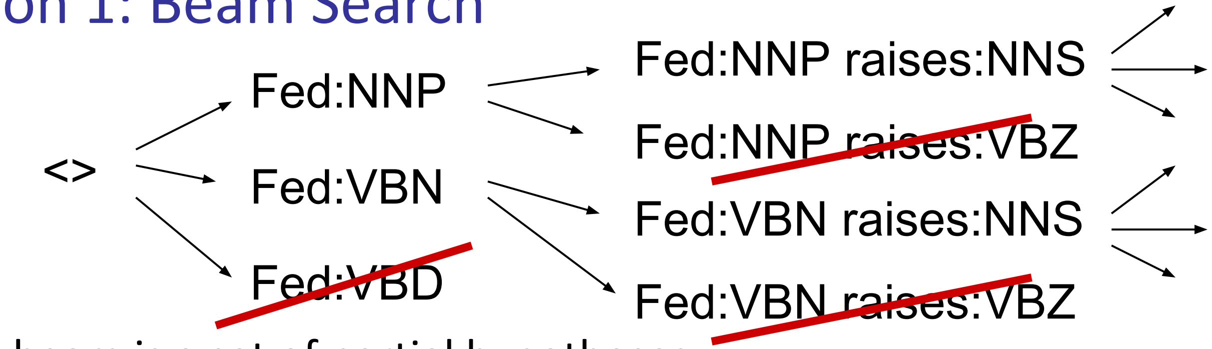
NNP VBZ NN NNS CD NN $\Rightarrow$ logP = -23

NNP NNS NN NNS CD NN $\Rightarrow$ logP = -29

NNP VBZ VB NNS CD NN $\Rightarrow$ logP = -27

Finding the Best Trajectory

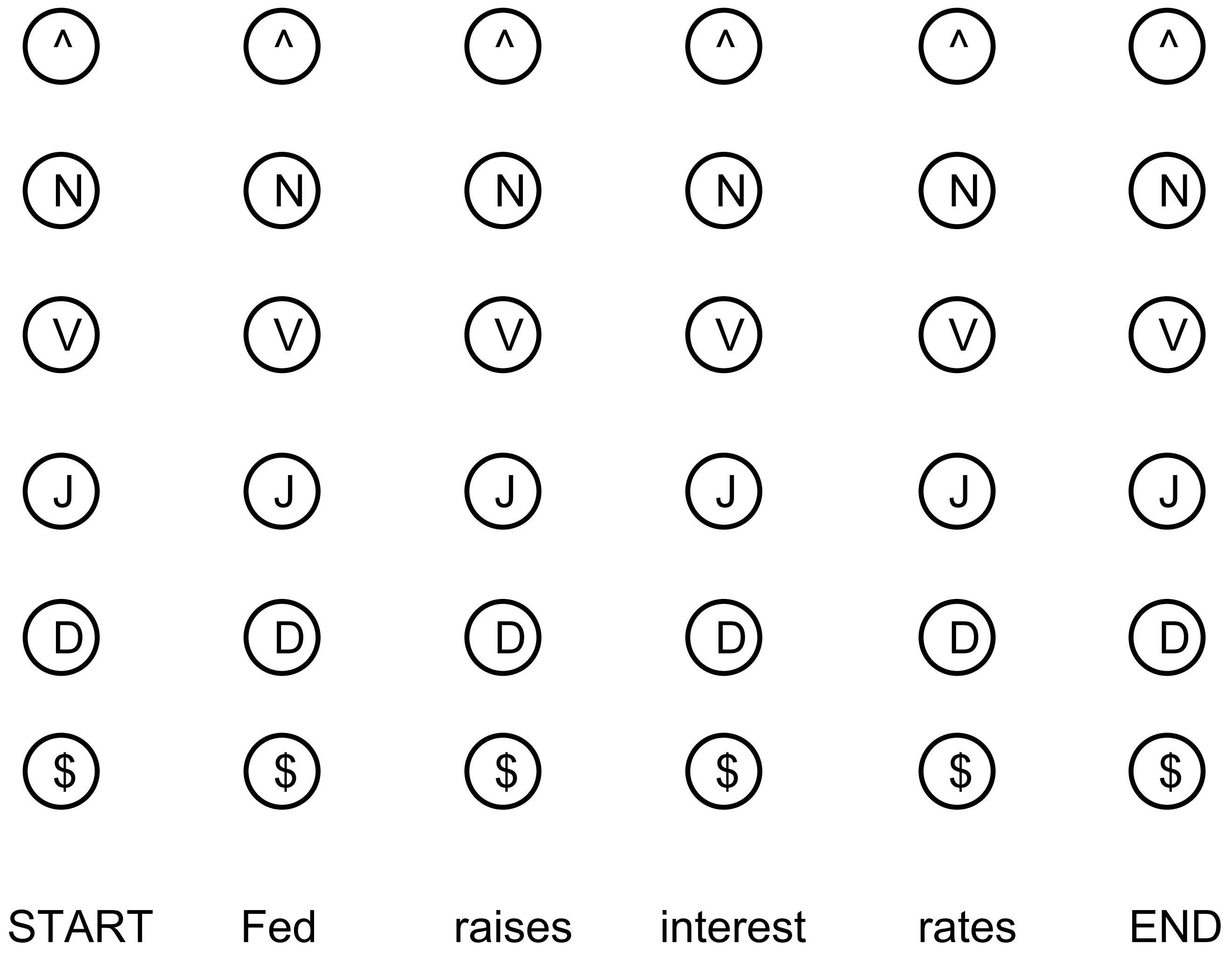
- Too many trajectories (state sequences) to list
- Option 1: Beam Search



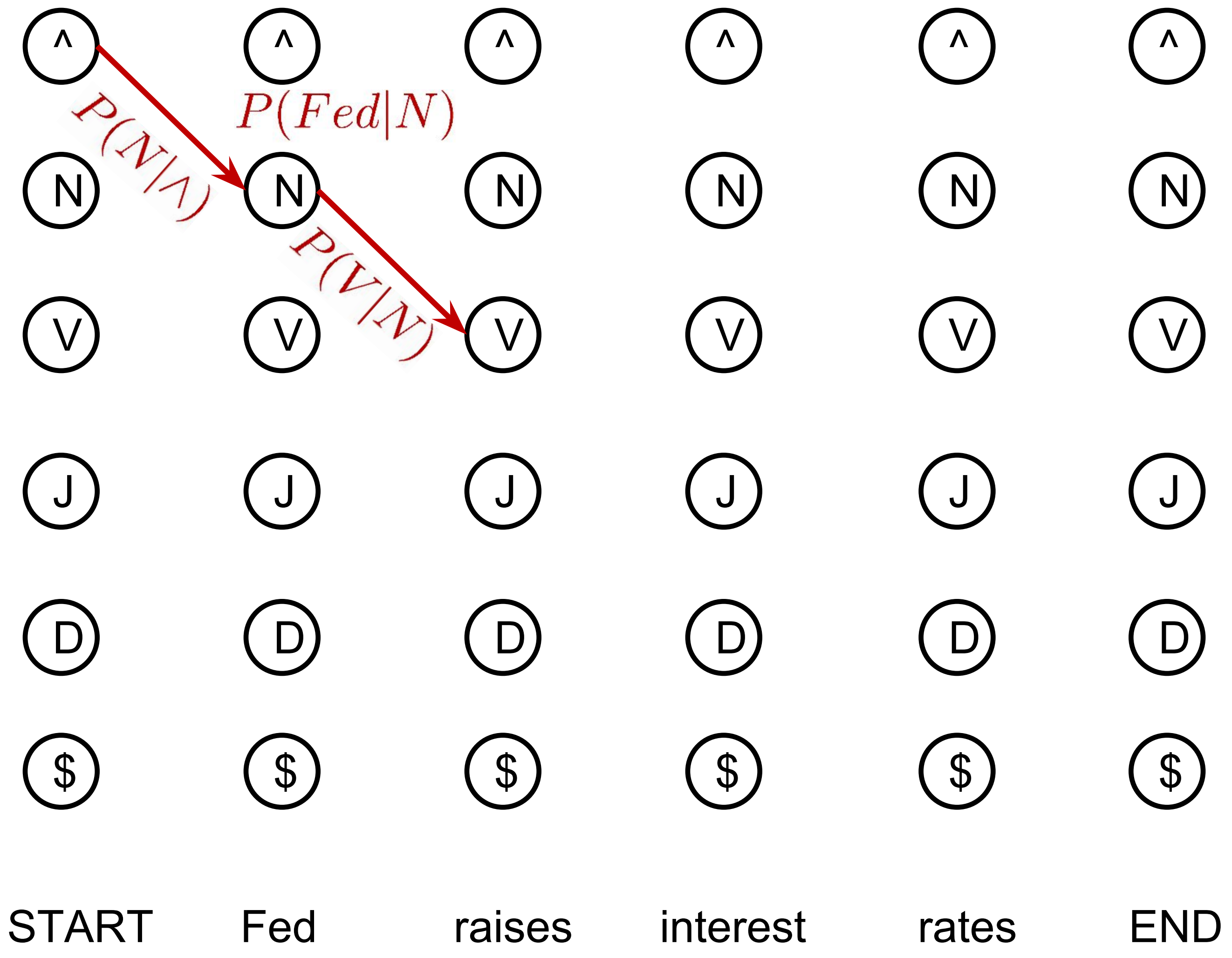
  - A beam is a set of partial hypotheses
  - Start with just the single empty trajectory
  - At each derivation step:
    - Consider all continuations of previous hypotheses
    - Discard most, keep top k, or those within a factor of the best

- Beam search works ok in practice
  - … but sometimes you want the optimal answer
  - … and you need optimal answers to validate your beam search
  - … and there's usually a better option than naïve beams

The State Lattice / Trellis



The State Lattice / Trellis



The Viterbi Algorithm

- Dynamic program for computing

$$\delta_i(s) = \max_{s_0 \ldots s_{i-1} s} P(s_0 \ldots s_{i-1} s, w_1 \ldots w_{i-1})$$

  - The score of a best path up to position i ending in state s

$$\delta_0(s) = \begin{cases} 1 & if \;\; s = <\bullet,\bullet> \\ 0 & otherwise \end{cases}$$

$$\delta_i(s) = \max_{s'} P(s \mid s')P(w \mid s')\delta_{i-1}(s')$$

  - Also can store a backtrace

$$\psi_i(s) = \arg\max_{s'} P(s \mid s')P(w \mid s')\delta_{i-1}(s')$$

- Memoized solution
- Iterative solution

So How Well Does It Work?

- Choose the most common tag
  - 90.3% with a bad unknown word model
  - 93.7% with a good one
- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text (SOTA is 97+%)
- Noise in the data
  - Many errors in the training and test corpora

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

  - 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%
  - TnT (HMM++): 96.5% / 85.9%
  - Maxent P(t|w): 93.7% / 82.6%
  - MEMM tagger: 96.9% / 86.9%
  - State-of-the-art: 97+% / 92+%
  - Upper bound: ~98%

Most errors are on unknown words

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/JJ NN
US Navy officer

VBD RP/IN DT NN
got around the corner

PRP RB VBD/VBN NNS IN …
They recently sold shares at …

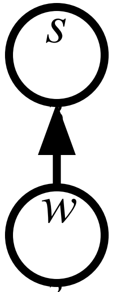
Richer Features

Better Features

- Can do surprisingly well just looking at a word by itself:

  - Word the: the → DT
  - Lowercased word Importantly: importantly → RB
  - Prefixes unfathomable: un- → JJ
  - Suffixes Surprisingly: -ly → RB
  - Capitalization Meridian: CAP → NNP
  - Word shapes 35-year: d-x → JJ

- Then build a maxent (or whatever) model to predict tag
- Maxent P(t|w): 93.7% / 82.6%



Why Linear Context is Useful

- Lots of rich local information!

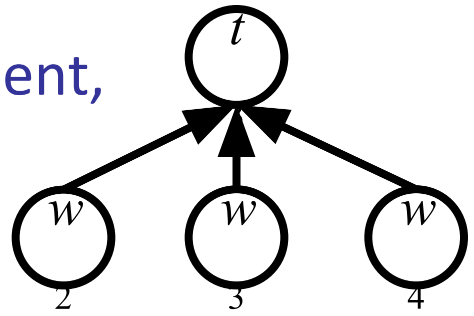
		RB					
PRP	VBD	IN	RB	IN	PRP	VBD	.
They	left	as	soon	as	he	arrived	.

  - We could fix this with a feature that looked at the next word

JJ				
NNP	NNS	VBD	VBN	.
Intrinsic	flaws	remained	undetected	.

  - We could fix this by linking capitalized words to their lowercase versions
- Solution: discriminative sequence models (MEMMs, CRFs)
- Reality check:
  - Taggers are already pretty good on newswire text…
  - What the world needs is taggers that work on other text!

Sequence-Free Tagging?



- 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
- All features but no sequence: 96.6% / 86.8%
- Uses lots of features: > 200K
- Why isn’t this the standard approach?

Named Entity Recognition

- Other sequence tasks use similar models

- Example: name entity recognition (NER)

PER PER O O O O O O ORG O O O O O LOC LOC O

Tim Boon has signed a contract extension with Leicestershire which will keep him at Grace Road .

Local Context

	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

MEMM Taggers

- Idea: left-to-right local decisions, condition on previous tags and also entire input

$$P(\mathbf{t}|\mathbf{w}) = \prod_i P_{\text{ME}}(t_i|\mathbf{w}, t_{i-1}, t_{i-2})$$

  - Train $P(t_i|w,t_{i-1},t_{i-2})$ as a normal maxent model, then use to score sequences
  - This is referred to as an MEMM tagger [Ratnaparkhi 96]
  - Beam search effective! (Why?)
  - What about beam size 1?

- Issues with local normalization, called "Label Bias Problem" (cf. Lafferty et al. 2001 and many others)
- Should no longer be used in practice (but neural models often do this)

NER Features

Because of regularization term, the more common prefixes can have larger weights even though entire-word features are more specific.

Local Context

	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Feature Weights

Feature Type	**Feature**	**PERS**	**LOC**
Previous word	*at*	-0.73	0.94
Current word	*Grace*	0.03	0.00
Beginning bigram	*<G*	0.45	-0.04
Current POS tag	NNP	0.47	0.45
Prev and cur tags	IN NNP	-0.10	0.14
Previous state	Other	-0.70	-0.92
Current signature	Xx	0.80	0.46
Prev state, cur sig	O-Xx	0.68	0.37
Prev-cur-next sig	x-Xx-Xx	-0.69	0.37
P. state - p-cur sig	O-x-Xx	-0.20	0.82
…			
Total:		**-0.58**	**2.68**

Decoding

- Decoding MEMM taggers:
  - Just like decoding HMMs, different local scores
  - Viterbi, beam search, posterior decoding
- Viterbi algorithm (HMMs):

$$\delta_i(s) = \arg\max_{s'} P(s|s')P(w_{i-1}|s')\delta_{i-1}(s')$$

- Viterbi algorithm (MEMMs):

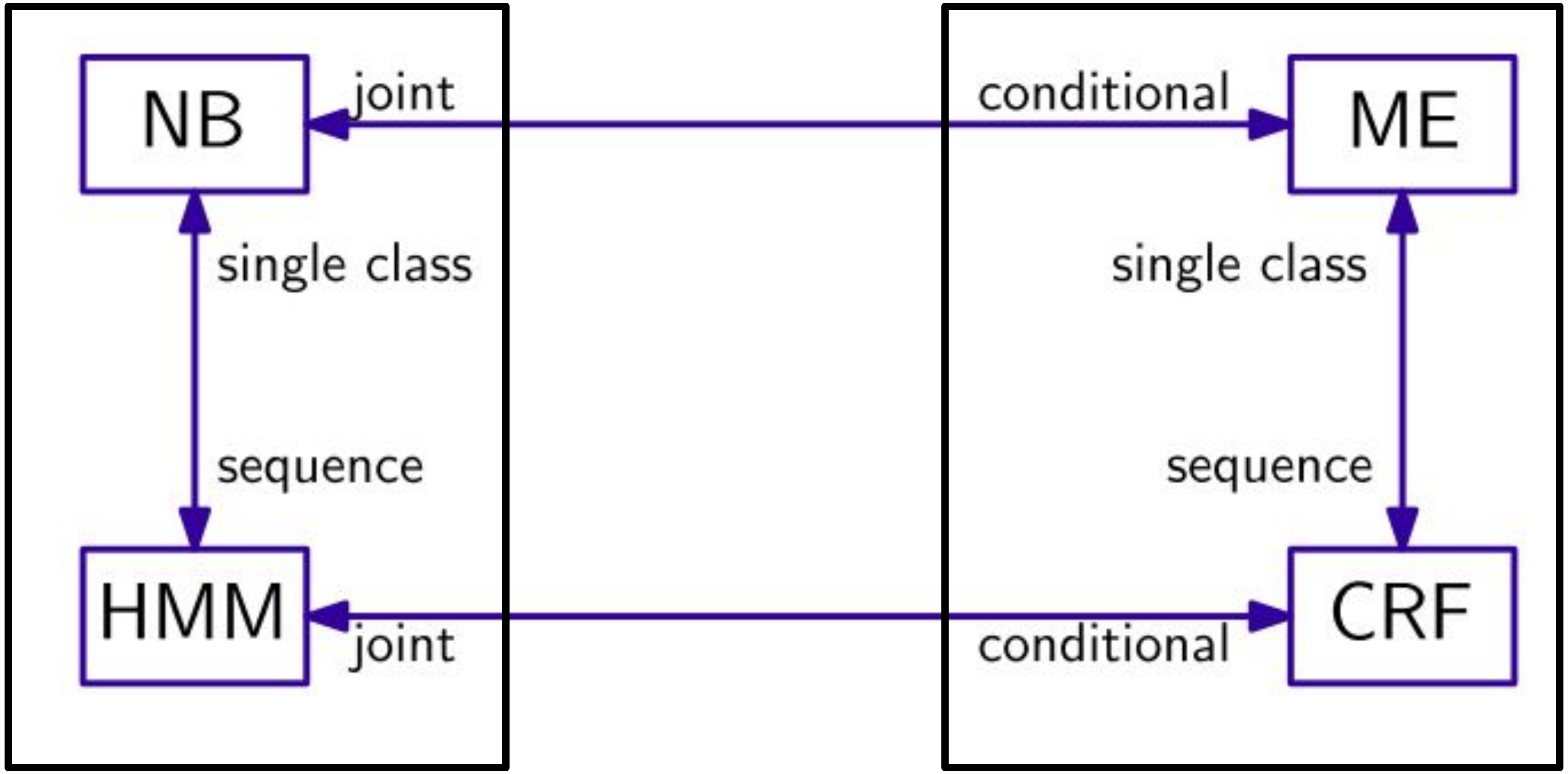
$$\delta_i(s) = \arg\max_{s'} P(s|s', \mathbf{w})\delta_{i-1}(s')$$

- General:

$$\delta_i(s) = \arg\max_{s'} \phi_i(s', s)\delta_{i-1}(s')$$

Conditional Random Fields
(and Friends)

HMM and Other Probabilistic Models



Generative
P(x|y)

Discriminative
P(y|x)

Generative vs Discriminative

- Generative Models (P(x|y)):
  - Try to approximate the process that generates the observation
    - E.g., the topic of a document determines the distribution of the words
  - Turns out this can be used for classification as well by inverting the probability
  - Examples: Naïve Bayes, HMM, GMM, VAE
  - Focus: modeling data generation process
- Discriminative Models (P(y|x)):
  - If we are interested just in the tags/classes, try to model it directly
    - E.g., the words present in a document distinguish (discriminate) between different topics
  - Examples: Perceptron, MEMM, SVM, CRF
  - Focus: creating decision boundaries

[Collins 01]

Perceptron Taggers

- Linear models:

$$\text{score}(\mathbf{t}|\mathbf{w}) = \lambda^\top f(\mathbf{t}, \mathbf{w})$$

- … that decompose along the sequence

$$= \lambda^\top \sum_i f(t_i, t_{i-1}, \mathbf{w}, i)$$

- … allow us to predict with the Viterbi algorithm

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} \ \text{score}(\mathbf{t}|\mathbf{w})$$

- … which means we can train with the perceptron algorithm (or related updates, like MIRA)

Perceptron Algorithm

Algorithm 1 Multi-class Perceptron algorithm

Require: Training data: $\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|T|}$

$\mathbf{w} = \mathbf{0}$ ▷ or small random vectors

for $n := 1$ **to** N **do** ▷ number of iterations

  for $t := 1$ **to** $|T|$ **do** ▷ number of training instances

    $\mathbf{y}' = \text{argmax}_{\mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y})$ ▷ best output

    if $\mathbf{y}' \neq \mathbf{y}_t$ **then** ▷ if incorrect

      $\mathbf{w} = \mathbf{w} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t + \mathbf{y}')$ ▷ update

    end if

  end for

end for

Transformation-Based Learning

- [Brill 95] presents a *transformation-based* tagger
  - Label the training set with most frequent tags

    DT MD VBD VBD .
    The can was rusted .

  - Add transformation rules which reduce training mistakes
    - MD $\rightarrow$ NN : DT __
    - VBD $\rightarrow$ VBN : VBD __ .
  - Stop when no transformations do sufficient good
  - Does this remind anyone of anything?
- ~~Probably the most widely used tagger (esp. outside NLP)~~
- … but definitely not the most accurate: 96.6% / 82.0 %

Learned Transformations

- What gets learned? [from Brill 95]

#	Change Tag From	Change Tag To	Condition
1	NN	VB	Previous tag is *TO*
2	VBP	VB	One of the previous three tags is *MD*
3	NN	VB	One of the previous two tags is *MD*
4	VB	NN	One of the previous two tags is *DT*
5	VBD	VBN	One of the previous three tags is *VBZ*
6	VBN	VBD	Previous tag is *PRP*
7	VBN	VBD	Previous tag is *NNP*
8	VBD	VBN	Previous tag is *VBD*
9	VBP	VB	Previous tag is *TO*
10	POS	VBZ	Previous tag is *PRP*
11	VB	VBP	Previous tag is *NNS*
12	VBD	VBN	One of previous three tags is *VBP*
13	IN	WDT	One of next two tags is *VB*
14	VBD	VBN	One of previous two tags is *VB*
15	VB	VBP	Previous tag is *PRP*
16	IN	WDT	Next tag is *VBZ*
17	IN	DT	Next tag is *NN*
18	JJ	NNP	Next tag is *NNP*
19	IN	WDT	Next tag is *VBD*
20	JJR	RBR	Next tag is *JJ*

#	Change Tag From	Change Tag To	Condition
1	NN	NNS	Has suffix **-s**
2	NN	CD	Has character **.**
3	NN	JJ	Has character **-**
4	NN	VBN	Has suffix **-ed**
5	NN	VBG	Has suffix **-ing**
6	??	RB	Has suffix **-ly**
7	??	JJ	Adding suffix **-ly** results in a word.
8	NN	CD	The word **$** can appear to the left.
9	NN	JJ	Has suffix **-al**
10	NN	VB	The word **would** can appear to the left.
11	NN	CD	Has character **0**
12	NN	JJ	The word **be** can appear to the left.
13	NNS	JJ	Has suffix **-us**
14	NNS	VBZ	The word **it** can appear to the left.
15	NN	JJ	Has suffix **-ble**
16	NN	JJ	Has suffix **-ic**
17	NN	CD	Has character **1**
18	NNS	NN	Has suffix **-ss**
19	??	JJ	Deleting the prefix **un-** results in a word
20	NN	JJ	Has suffix **-ive**

Maximum Entropy++

- Remember: maximum entropy objective

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

- Problem: lots of features allow perfect fit to training set
- Regularization (compare to smoothing)

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

Derivative for Maximum Entropy

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

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

Big weights are bad

Expected count of feature n in predicted candidates

Total count of feature n in correct candidates

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 tend not to be too important for POS tagging
- Differences more substantial on other sequence tasks
- 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

Conditional Random Fields (CRFs)

- CRF: Make a maxent model over entire taggings
  - MEMM

$$P(\mathbf{t}|\mathbf{w}) = \prod_i \frac{1}{Z(i)} \exp\left(\lambda^\top f(t_i, t_{i-1}, \mathbf{w}, i)\right)$$

  - CRF

$$P(\mathbf{t}|\mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left(\lambda^\top f(\mathbf{t}, \mathbf{w})\right)$$

$$= \frac{1}{Z(\mathbf{w})} \exp\left(\lambda^\top \sum_i f(t_i, t_{i-1}, \mathbf{w}, i)\right)$$

$$= \frac{1}{Z(\mathbf{w})} \prod_i \phi_i(t_i, t_{i-1})$$

Conditional Random Fields (CRFs)

- Like any maxent model, derivative is:

$$\frac{\partial L(\lambda)}{\partial \lambda} = \sum_k \left(\mathbf{f}_k(\mathbf{t}^k) - \sum_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}_k)\mathbf{f}_k(\mathbf{t}) \right)$$

- So all we need is to be able to compute the expectation of each feature (for example the number of times the label pair *DT-NN* occurs, or the number of times *NN-interest* occurs) **under the model distribution**
- Critical quantity: counts of posterior marginals:

$$\text{count}(w, s) = \sum_{i:w_i=w} P(t_i = s|\mathbf{w})$$

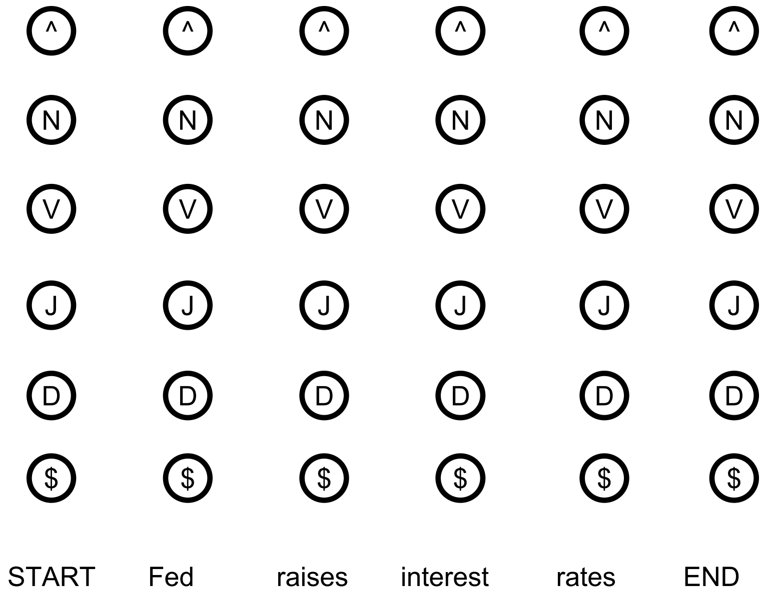
$$\text{count}(s \to s') = \sum_i P(t_{i-1} = s, t_i = s'|\mathbf{w})$$

Computing Posterior Marginals

- How many (expected) times is word w tagged with s?

$$\text{count}(w,s) = \sum_{i:w_i=w} P(t_i = s|\mathbf{w})$$

- How to compute that marginal?

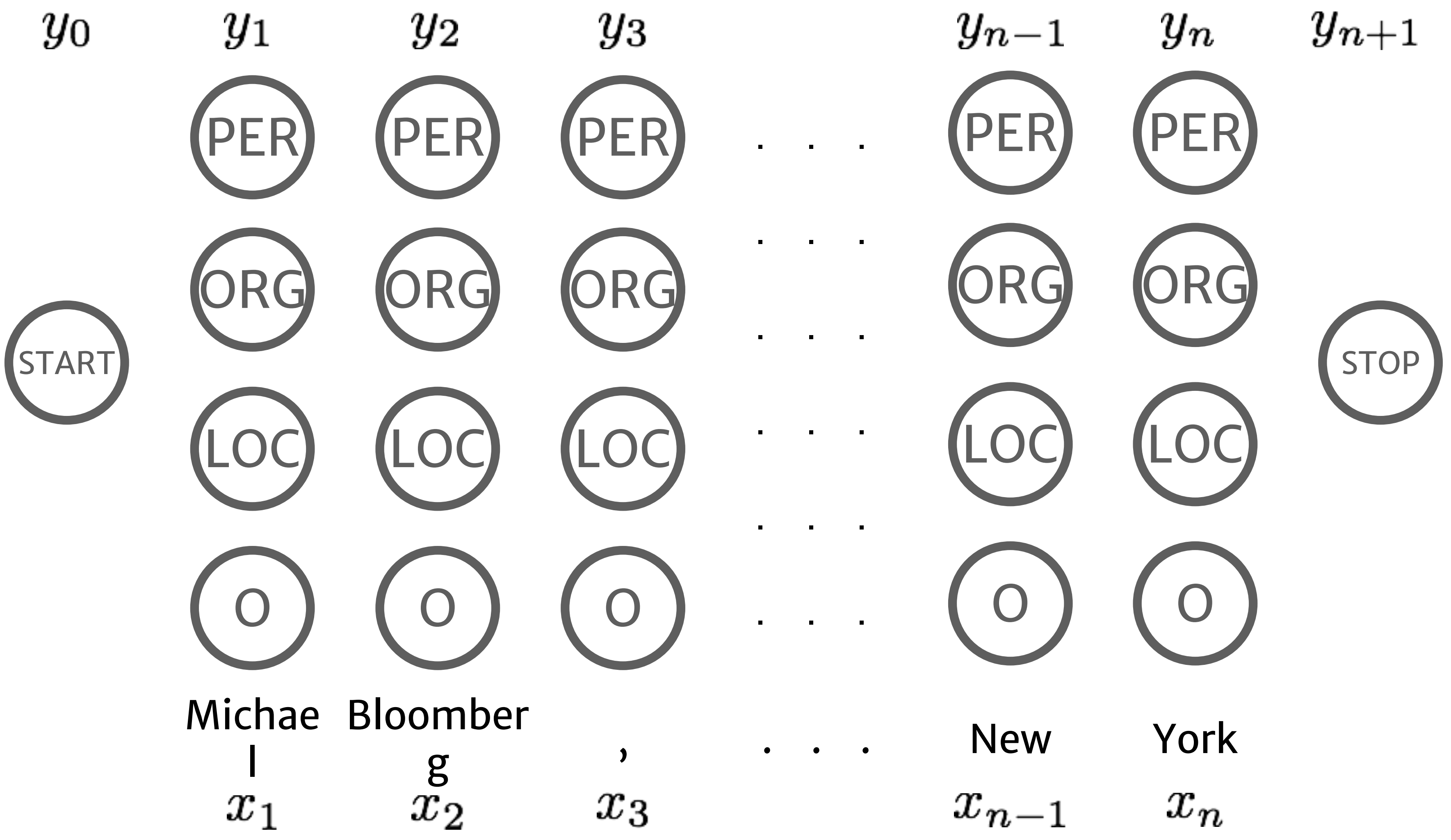


$$\alpha_i(s) = \sum_{s'} \phi_i(s', s)\alpha_{i-1}(s')$$

$$\beta_i(s) = \sum_{s'} \phi_{i+1}(s, s')\beta_{i+1}(s')$$

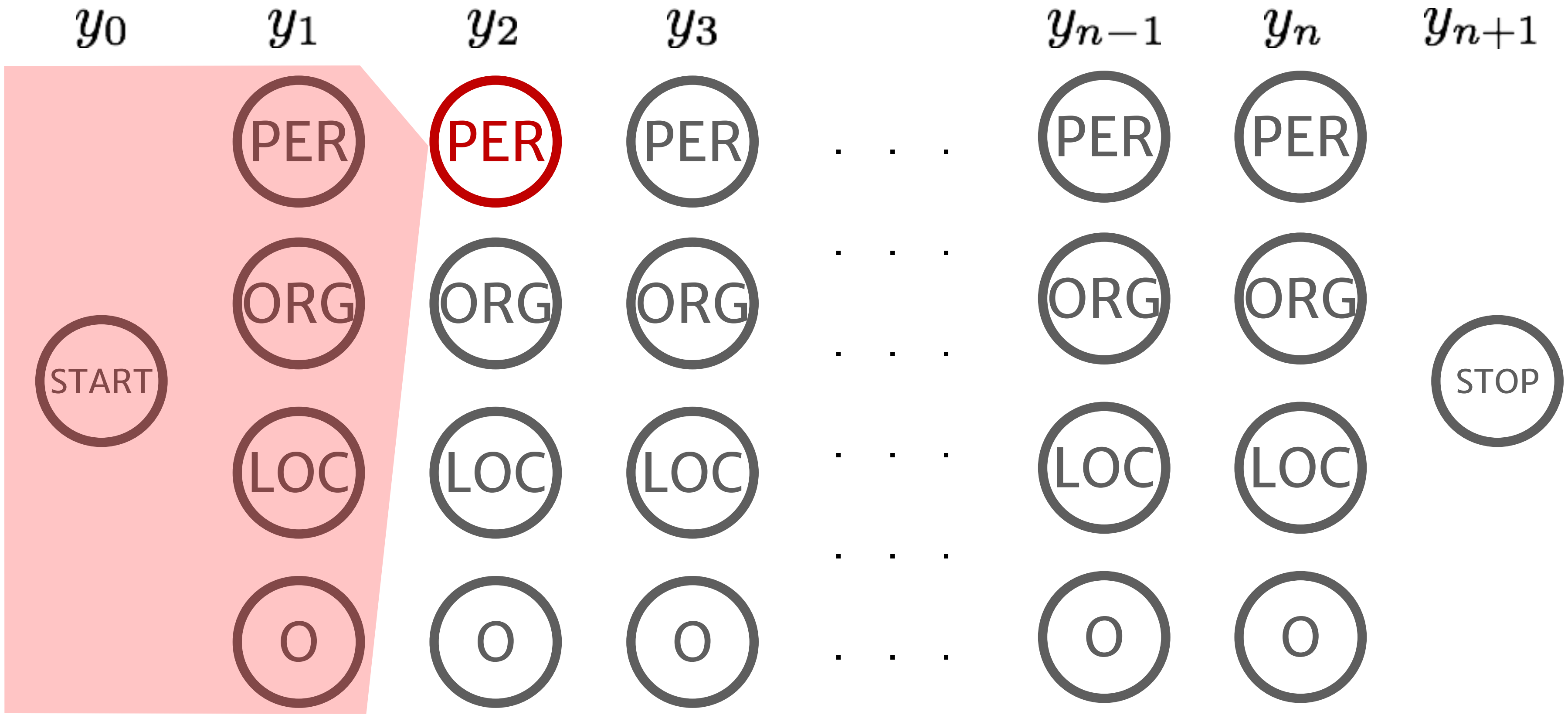
$$P(t_i = s|\mathbf{w}) = \frac{\alpha_i(s)\beta_i(s)}{\alpha_N(\text{END})}$$

Dynamic programming



[slide credit: Maria Ryskina]

Forward pass

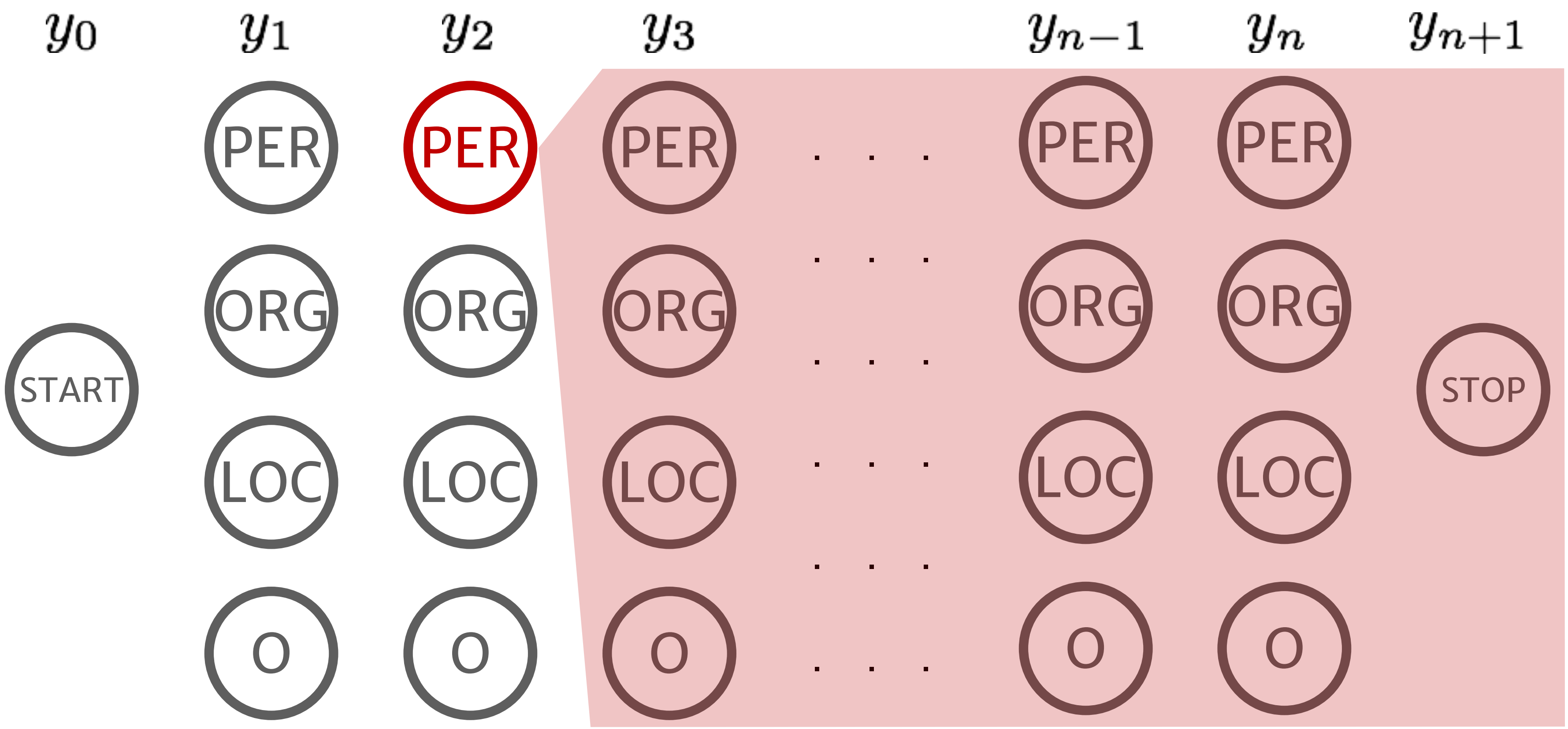


$$\alpha_t(s) = \sum_{s'} \exp(w^\top f(x, y_t = s, y_{t-1} = s'))\alpha_{t-1}(s')$$

$$\alpha_0(s) = \mathbb{1}[s = \text{START}]$$

[slide credit: Maria Ryskina]

Backward pass

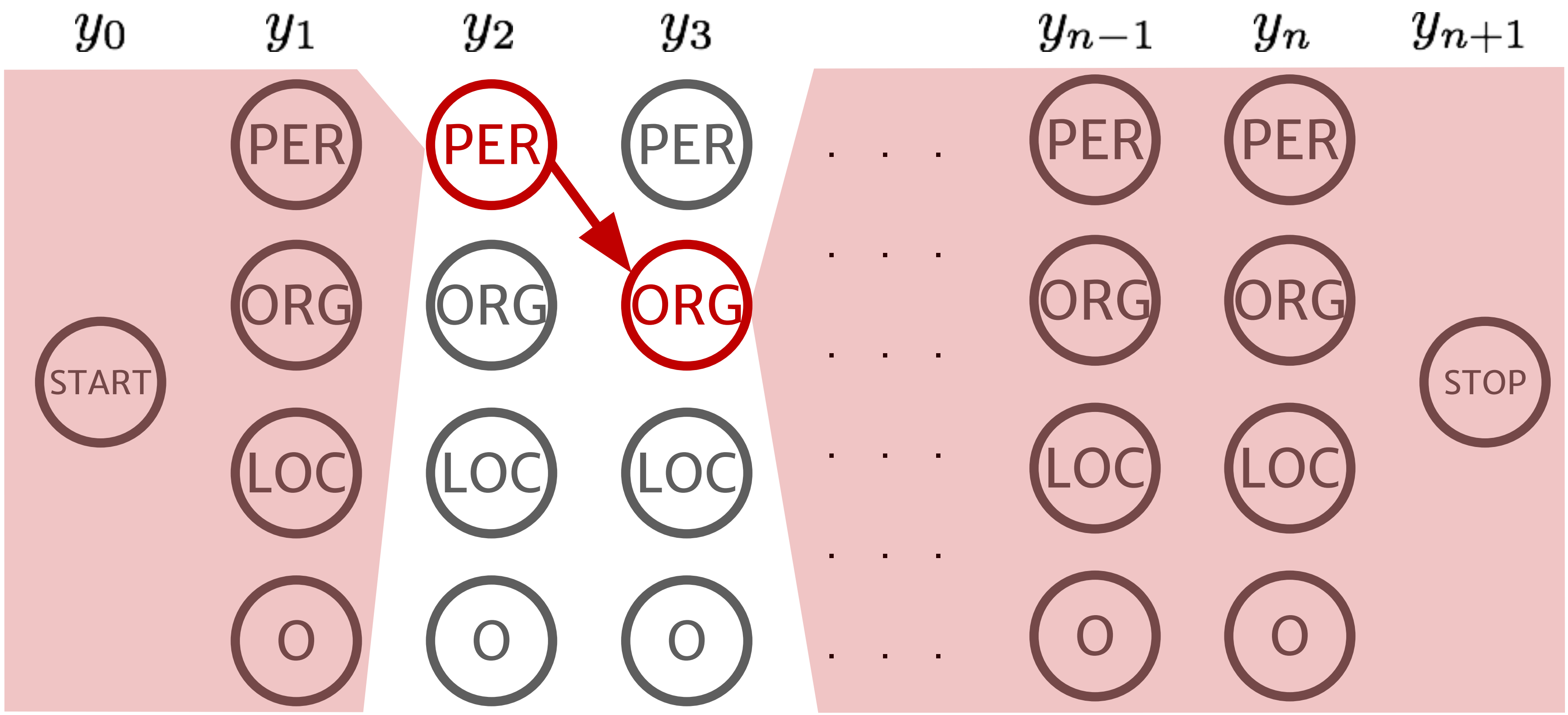


$$\beta_t(s) = \sum_{s'} \exp(w^\top f(x, y_t = s, y_{t+1} = s'))\beta_{t+1}(s')$$

$$\beta_{n+1}(s) = \mathbb{1}[s = \text{STOP}]$$

[slide credit: Maria Ryskina]

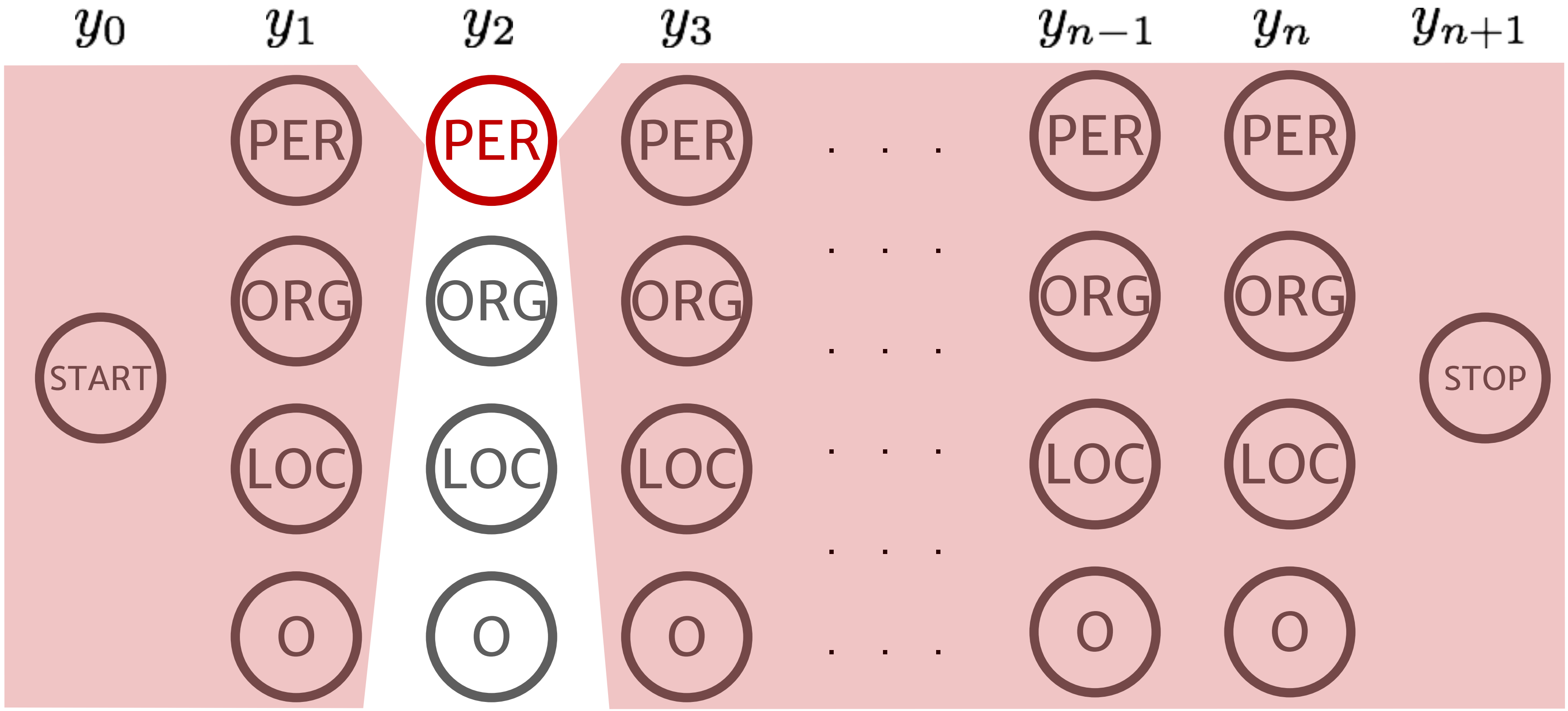
Computing marginals



$$P(y_t = s, y_{t-1} = s' | x, w) = \frac{\alpha_{t-1}(s') \exp(w^\top f(x, y_t = s, y_{t-1} = s')) \beta_t(s)}{\alpha_{n+1}(\text{STOP})}$$

[slide credit: Maria Ryskina]

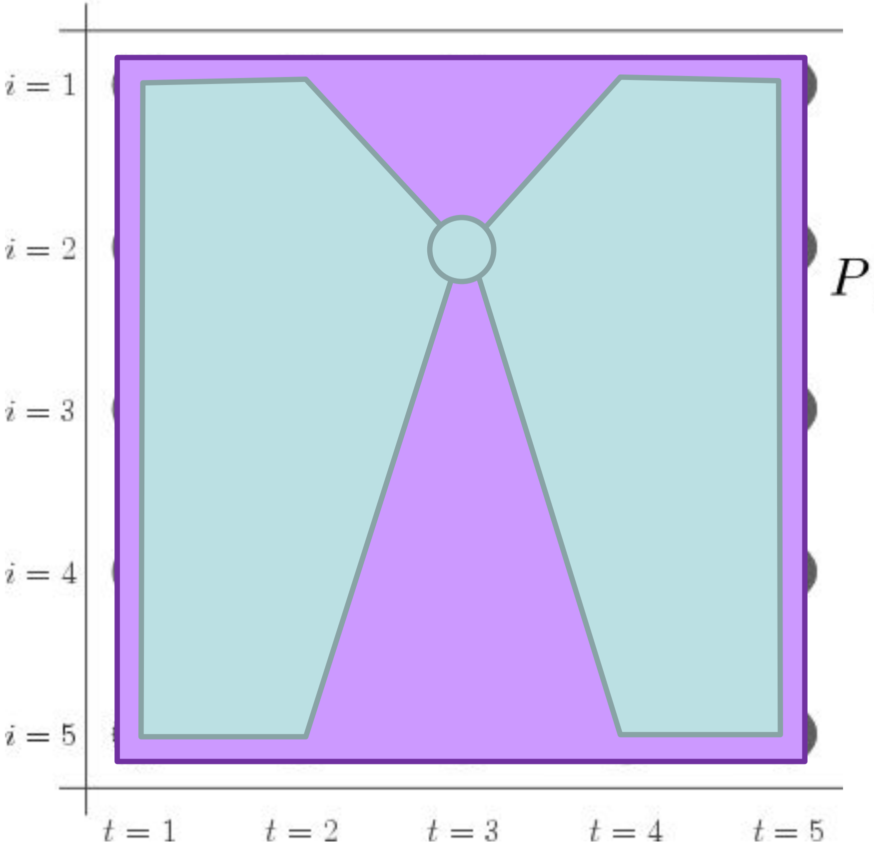
Computing marginals



$$P(y_t = s | x, w) = \frac{\alpha_t(s)\beta_t(s)}{\alpha_{n+1}(\text{STOP})}$$

[slide credit: Maria Ryskina]

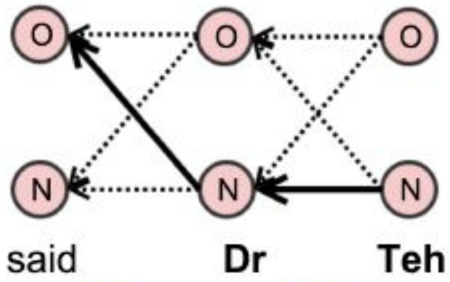
Forward-Backward



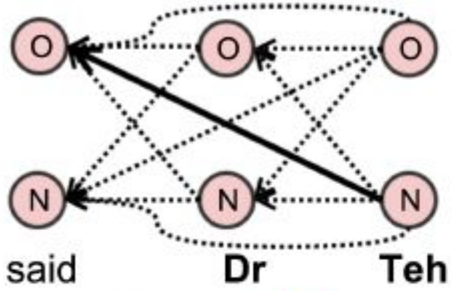
$$P(s_t|x) = \frac{P(s_t, x)}{P(x)}$$

$$= \frac{\text{sum of all paths through s at t}}{\text{sum of all possible paths}}$$

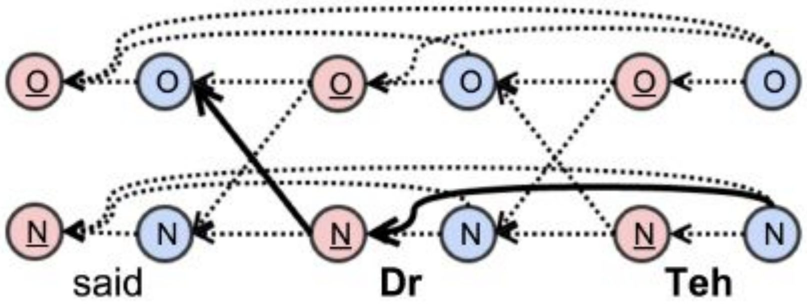
Variants of “All possible paths”



Linear CRF



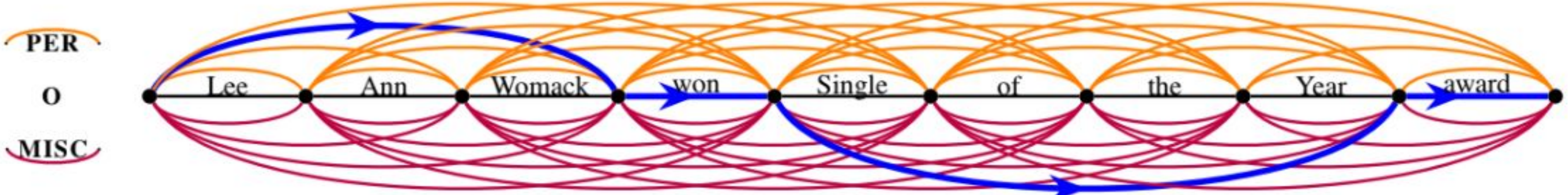
Semi-Markov CRF



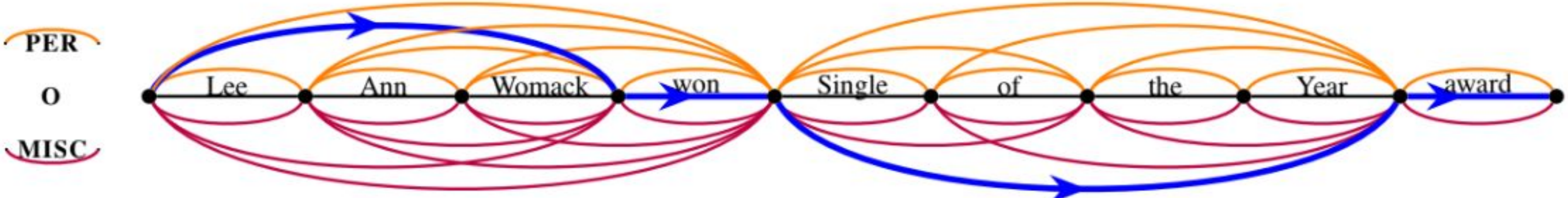
Weak Semi-Markov CRF

[Muis and Lu. 2016. Weak Semi-Markov CRFs for Noun Phrase Chunking in Informal Text. NAACL]

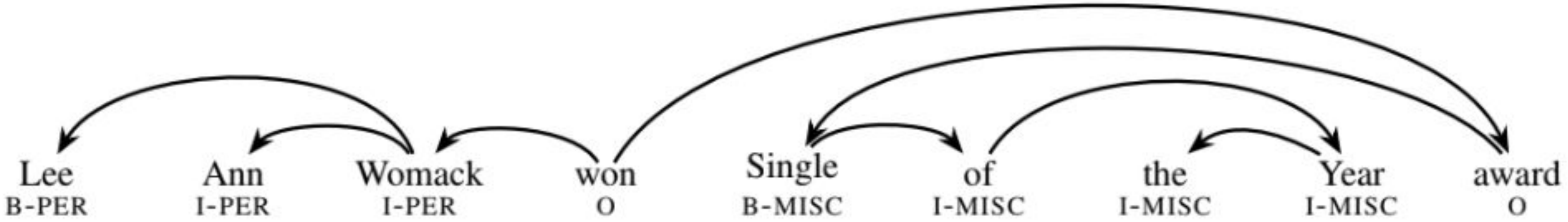
Variants of “All possible paths”



Semi-Markov CRF



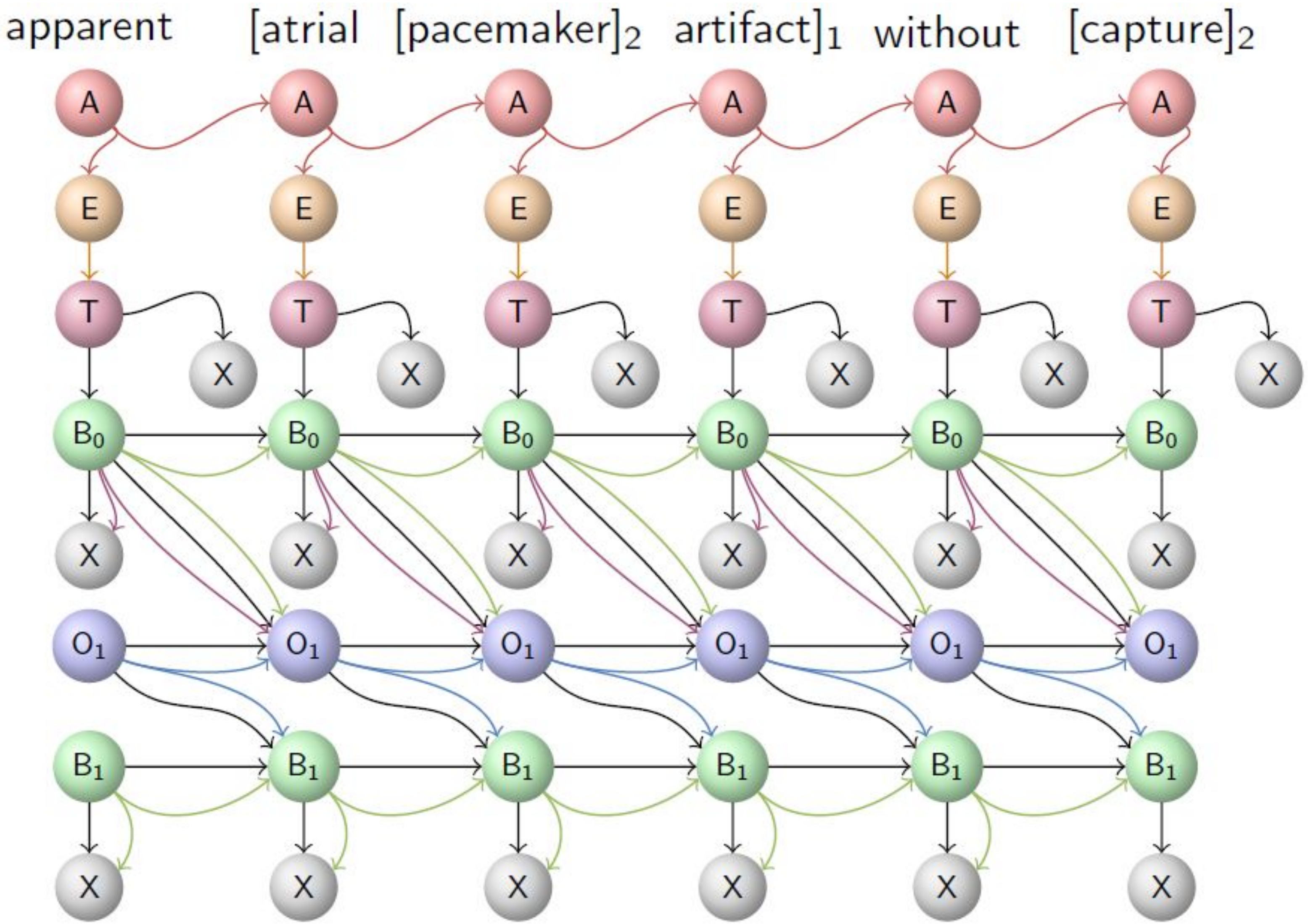
Dependency-guided CRF



Dependency tree

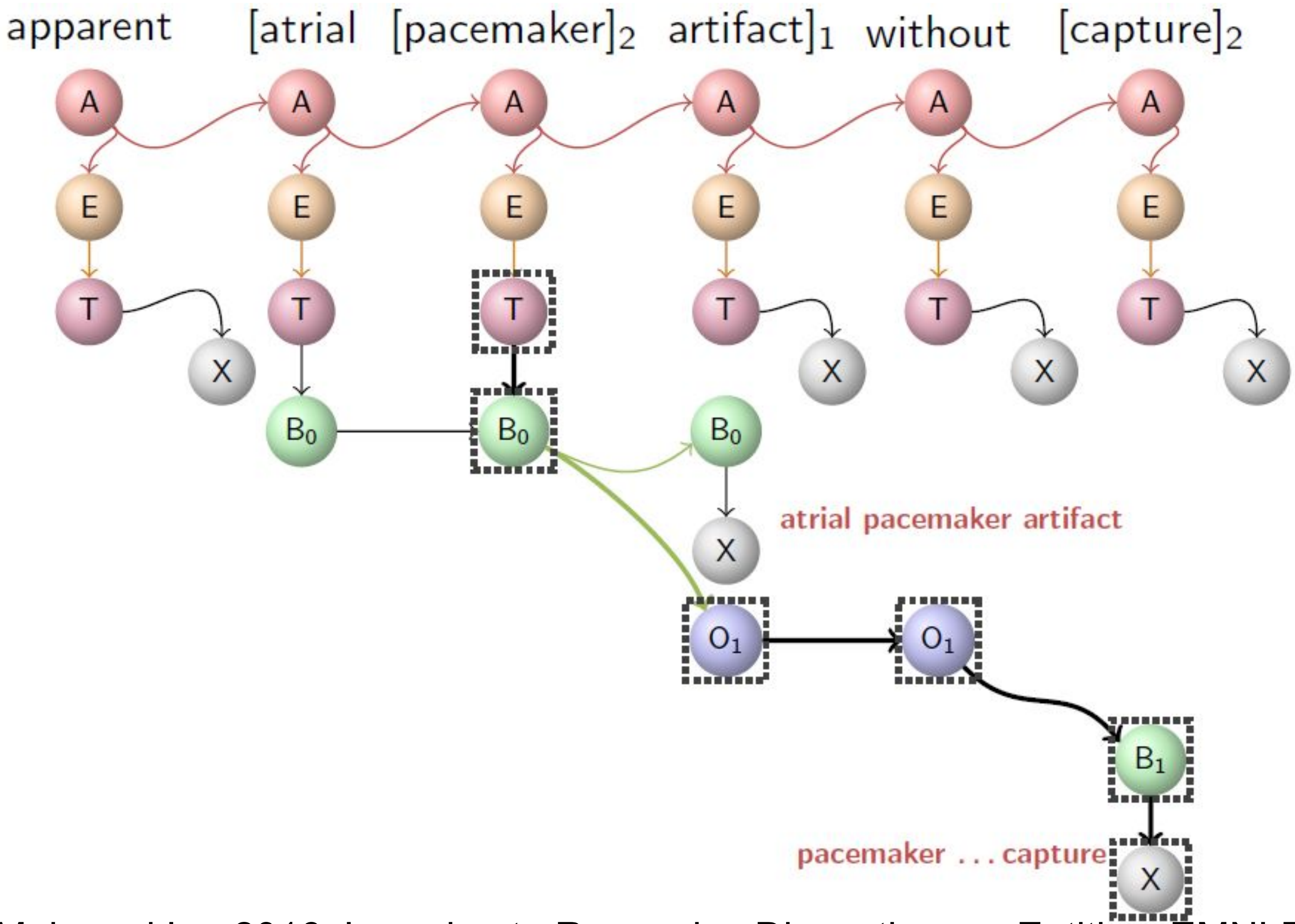
[Jie, Muis and Lu. 2017. Efficient Dependency-guided Named Entity Recognition. AAAI]

Variants of “All possible paths”



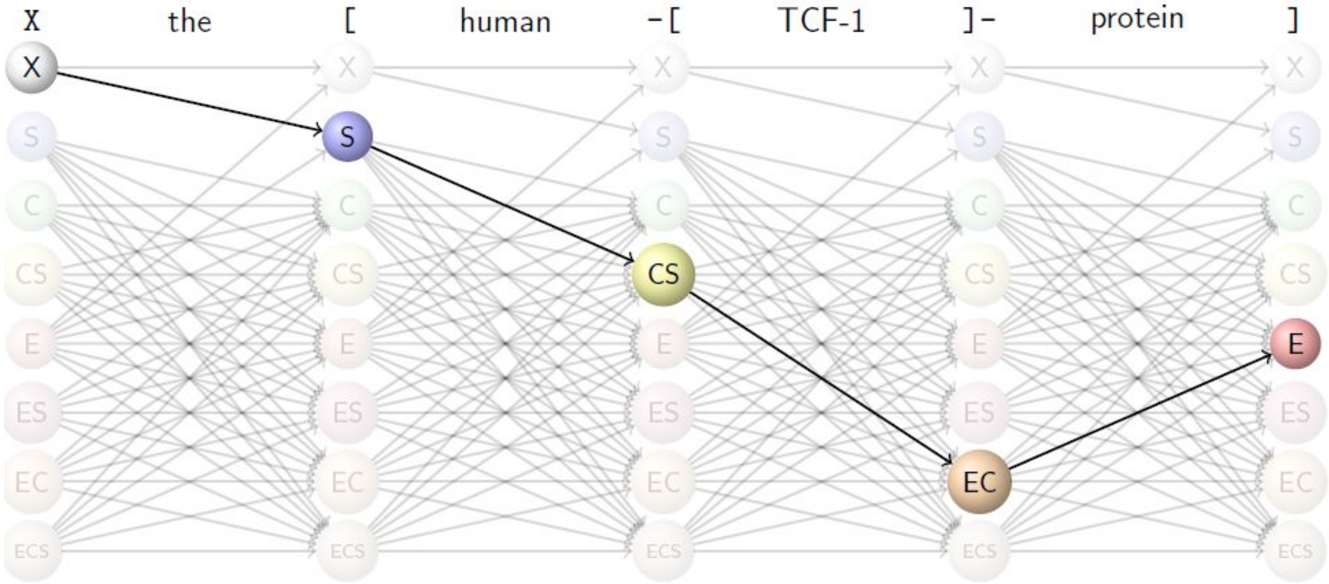
[Muis and Lu. 2016. Learning to Recognize Discontiguous Entities. EMNLP]

Variants of "All possible paths"

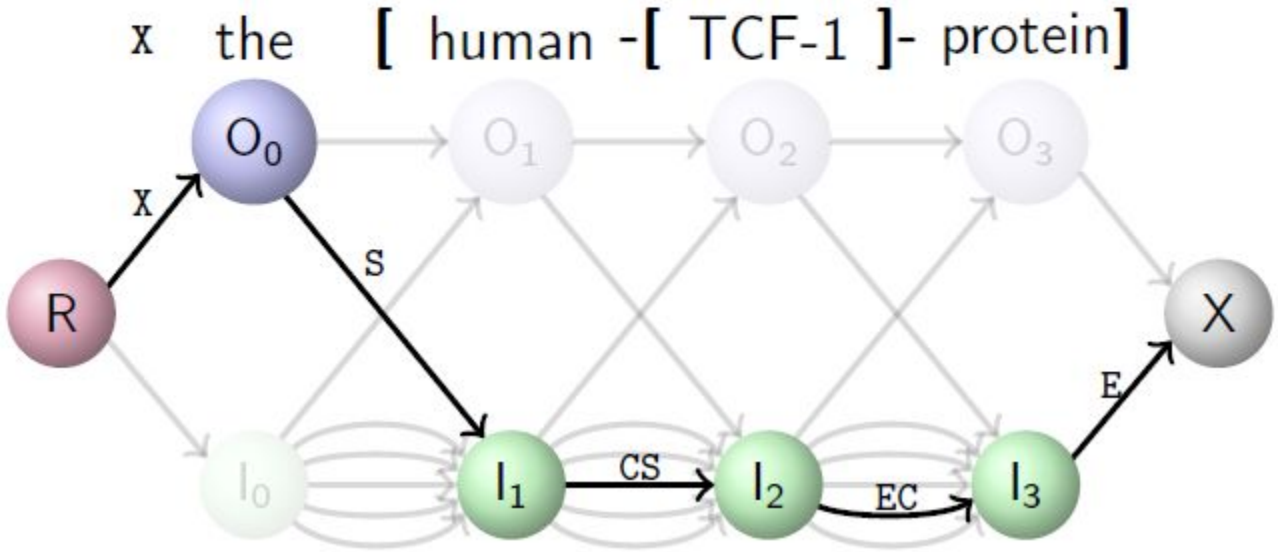


[Muis and Lu. 2016. Learning to Recognize Discontiguous Entities. EMNLP]

Variants of "All possible paths"



Linear-CRF:
8 states per word
8 x 8 = 64 edges between words



Multigraph CRF
2 states
8 edges between words

[Muis and Lu. 2017. Labeling Gaps between Words: Recognizing Overlapping Mentions with Mention Separators. EMNLP]

Variants on Objective Functions

$$\text{Structured Perceptron: } \min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{T}} -\overbrace{\mathbf{w}\cdot\Phi(x,y)}^{\text{Gold Score}} + \overbrace{\max_{\hat{y}\in\mathcal{Y}}}^{\text{Selector}} \Big(\overbrace{\mathbf{w}\cdot\Phi(x,\hat{y})}^{\text{Other Score}} \quad \overbrace{\qquad\qquad}^{\text{Cost}}\Big)$$

$$\text{CRF: } \min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{T}} -\mathbf{w}\cdot\Phi(x,y) + \operatorname*{softmax}_{\hat{y}\in\mathcal{Y}} \big(\mathbf{w}\cdot\Phi(x,\hat{y})\big)$$

$$\text{Max-Margin (SSVM): } \min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{T}} -\mathbf{w}\cdot\Phi(x,y) + \max_{\hat{y}\in\mathcal{Y}} \big(\mathbf{w}\cdot\Phi(x,\hat{y}) + \text{cost}(y,\hat{y})\big)$$

$$\text{Softmax-Margin: } \min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{T}} -\mathbf{w}\cdot\Phi(x,y) + \operatorname*{softmax}_{\hat{y}\in\mathcal{Y}} \big(\mathbf{w}\cdot\Phi(x,\hat{y}) + \text{cost}(y,\hat{y})\big)$$

[see also Gimpel and Smith. 2010.
Softmax-Margin Training for Structured Log-linear Models]

EngCG Tagger

- English constraint grammar tagger
  - [Tapanainen and Voutilainen 94]
  - Something else you should know about
  - Hand-written and knowledge driven
  - “Don’t guess if you know” (general point about modeling more structure!)
  - Tag set doesn’t make all of the hard distinctions as the standard tag set (e.g. JJ/NN)
  - They get stellar accuracies: 99% on *their* tag set
  - Linguistic representation matters…
  - … but it’s easier to win when you make up the rules

```
walk
   walk <SV> <SVO> V SUBJUNCTIVE VFIN
   walk <SV> <SVO> V IMP VFIN
   walk <SV> <SVO> V INF
   walk <SV> <SVO> V PRES -SG3 VFIN
   walk N NOM SG
```

```
walk V-SUBJUNCTIVE V-IMP V-INF
     V-PRES-BASE N-NOM-SG
```

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/language (what could we gain?) [Daume III. 2007 (FEDA), Kim et al. 2016 (Neural FEDA), Chen et al. 2016 (Deep Averaging Networks), Muis et al. 2018 (Distant Supervision)]
  - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)

Unsupervised Tagging

Unsupervised Tagging?

- AKA part-of-speech induction
- Task:
  - Raw sentences in
  - Tagged sentences out
- Obvious thing to do:
  - Start with a (mostly) uniform HMM
  - Run EM (Expectation-Maximization)
  - Inspect results

EM for HMMs: Process

- Alternate between recomputing distributions over hidden variables (the tags) and re-estimating parameters
- Crucial step: we want to tally up how many (fractional) counts of each kind of transition and emission we have under current params:

$$\text{count}(w, s) = \sum_{i:w_i=w} P(t_i = s|\mathbf{w})$$

$$\text{count}(s \to s') = \sum_i P(t_{i-1} = s, t_i = s'|\mathbf{w})$$

- Same quantities we needed to train a CRF!

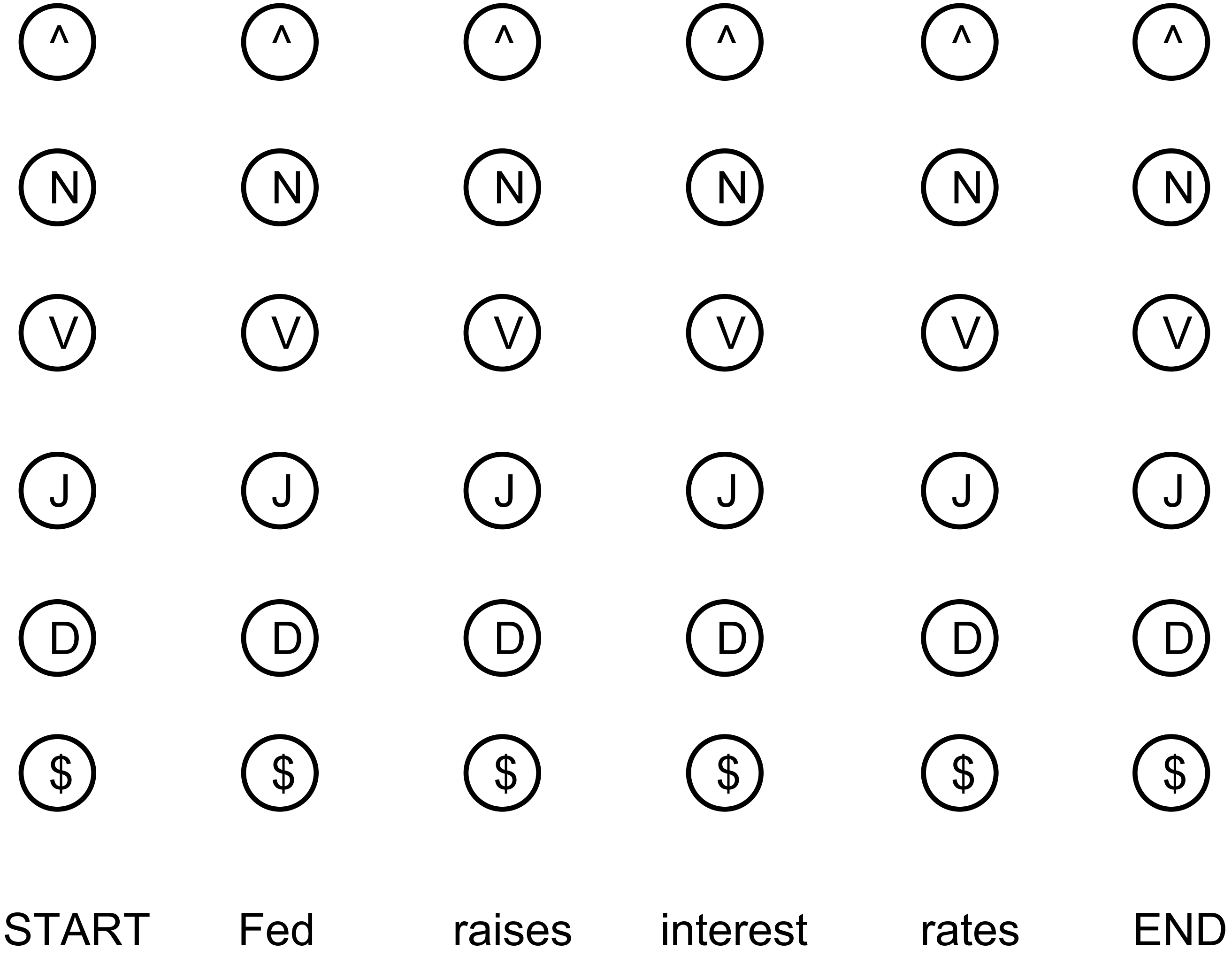
EM for HMMs: Quantities

- Total path values (correspond to probabilities here):

$$\begin{aligned}\alpha_i(s) &= P(w_0 \ldots w_i, s_i) \\ &= \sum_{s_{i-1}} P(s_i|s_{i-1})P(w_i|s_i)\alpha_{i-1}(s_{i-1})\end{aligned}$$

$$\begin{aligned}\beta_i(s) &= P(w_i + 1 \ldots w_n|s_i) \\ &= \sum_{s_{i+1}} P(s_{i+1}|s_i)P(w_{i+1}|s_{i+1})\beta_{i+1}(s_{i+1})\end{aligned}$$

The State Lattice / Trellis



EM for HMMs: Process

- From these quantities, can compute expected transitions:

$$\text{count}(s \to s') = \frac{\sum_i \alpha_i(s) P(s'|s) P(w_i|s) \beta_{i+1}(s')}{P(\mathbf{w})}$$

- And emissions:

$$\text{count}(w, s) = \frac{\sum_{i:w_i=w} \alpha_i(s) \beta_{i+1}(s)}{P(\mathbf{w})}$$

Merialdo: Setup

- Some (discouraging) experiments [Merialdo 94]

- Setup:
  - You know the set of allowable tags for each word
  - Fix k training examples to their true labels
    - Learn $P(w|t)$ on these examples
    - Learn $P(t|t_{-1},t_{-2})$ on these examples
  - On n examples, re-estimate with EM

- Note: we know allowed tags but not frequencies

Merialdo: Results

Number of tagged sentences used for the initial model							
	0	100	2000	5000	10000	20000	all
Iter	Correct tags (% words) after ML on 1M words						
0	77.0	90.0	95.4	96.2	96.6	96.9	97.0
1	80.5	92.6	95.8	96.3	96.6	96.7	96.8
2	81.8	93.0	95.7	96.1	96.3	96.4	96.4
3	83.0	93.1	95.4	95.8	96.1	96.2	96.2
4	84.0	93.0	95.2	95.5	95.8	96.0	96.0
5	84.8	92.9	95.1	95.4	95.6	95.8	95.8
6	85.3	92.8	94.9	95.2	95.5	95.6	95.7
7	85.8	92.8	94.7	95.1	95.3	95.5	95.5
8	86.1	92.7	94.6	95.0	95.2	95.4	95.4
9	86.3	92.6	94.5	94.9	95.1	95.3	95.3
10	86.6	92.6	94.4	94.8	95.0	95.2	95.2

"There is no data like more data" – Mercer, 1985

Distributional Clustering

♦ *the president said that the downtown was over* ♦

president	*the __ of*
president	*the __ said*
governor	*the __ of*
governor	*the __ appointed*
said	*sources __ ♦*
said	*president __ that*
reported	*sources __ ♦*

president
governor

the
a

said
reported

[Finch and Chater 92, Shuetze 93, many others]

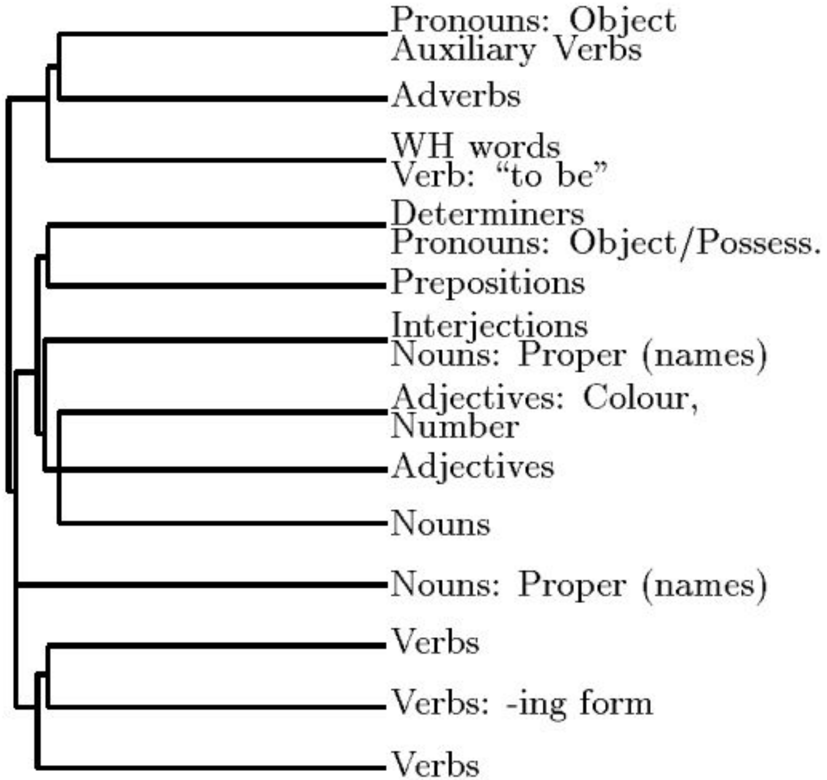
Distributional Clustering

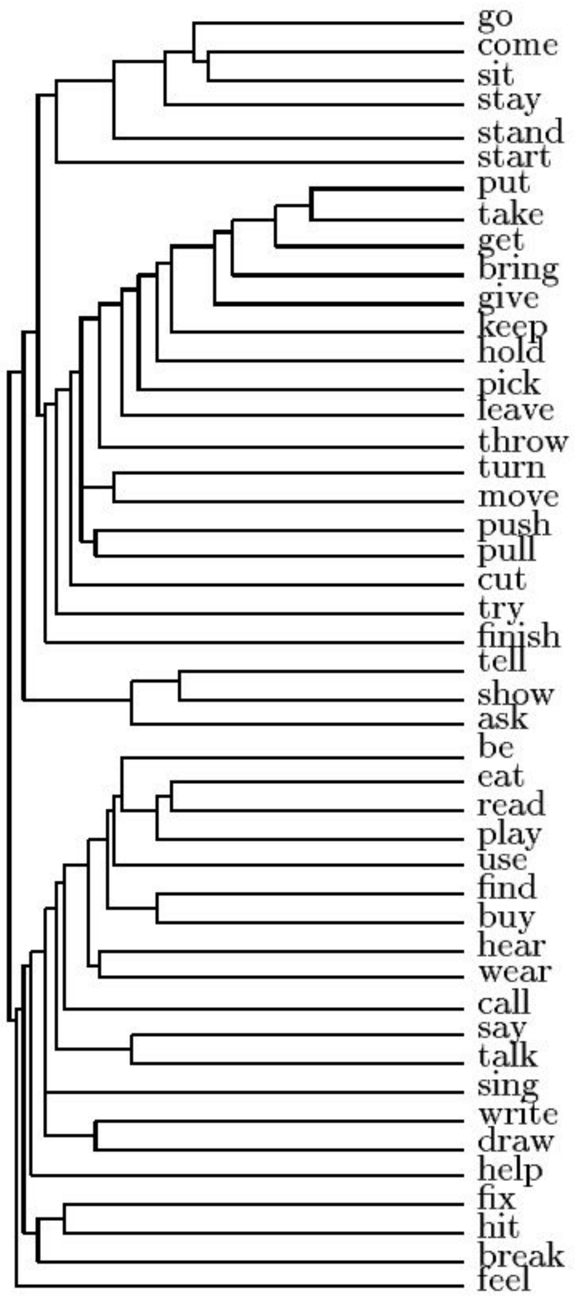
- Three main variants on the same idea:
  - Pairwise similarities and heuristic clustering
    - E.g. [Finch and Chater 92]
    - Produces dendrograms
  - Vector space methods
    - E.g. [Shuetze 93]
    - Models of ambiguity
  - Probabilistic methods
    - Various formulations, e.g. [Lee and Pereira 99]
  - Basis of (neural) word embedding

Nearest Neighbors

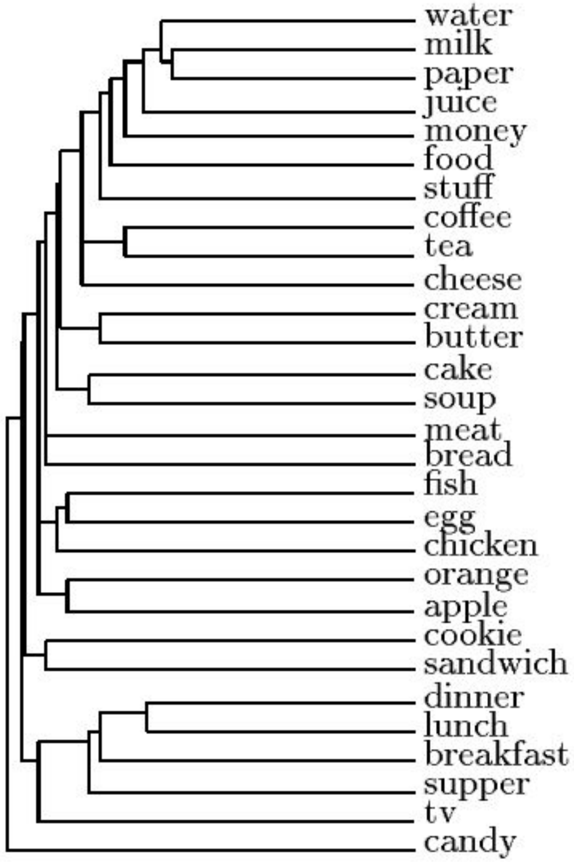
word	nearest neighbors
accompanied	submitted banned financed developed authorized headed canceled awarded barred
almost	virtually merely formally fully quite officially just nearly only less
causing	reflecting forcing providing creating producing becoming carrying particularly
classes	elections courses payments losses computers performances violations levels pictures
directors	professionals investigations materials competitors agreements papers transactions
goal	mood roof eye image tool song pool scene gap voice
japanese	chinese iraqi american western arab foreign european federal soviet indian
represent	reveal attend deliver reflect choose contain impose manage establish retain
think	believe wish know realize wonder assume feel say mean bet
york	angeles francisco sox rouge kong diego zone vegas inning layer
on	through in at over into with from for by across
must	might would could cannot will should can may does helps
they	we you i he she nobody who it everybody there

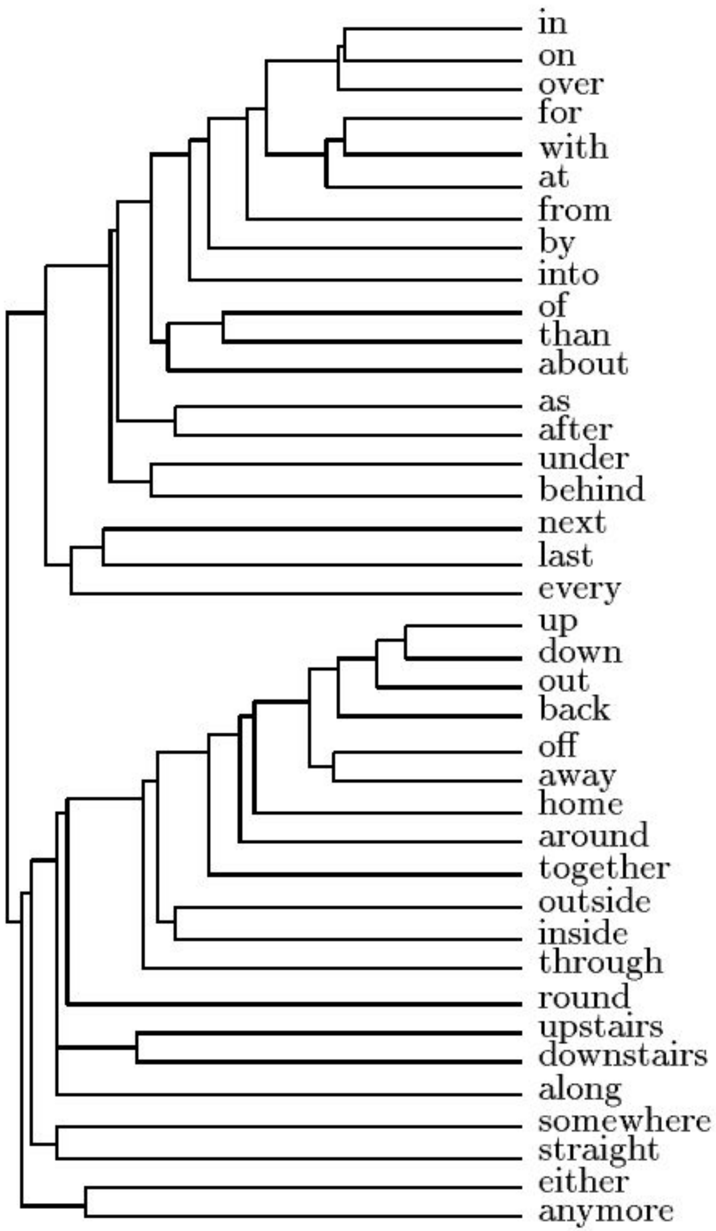
Dendrograms





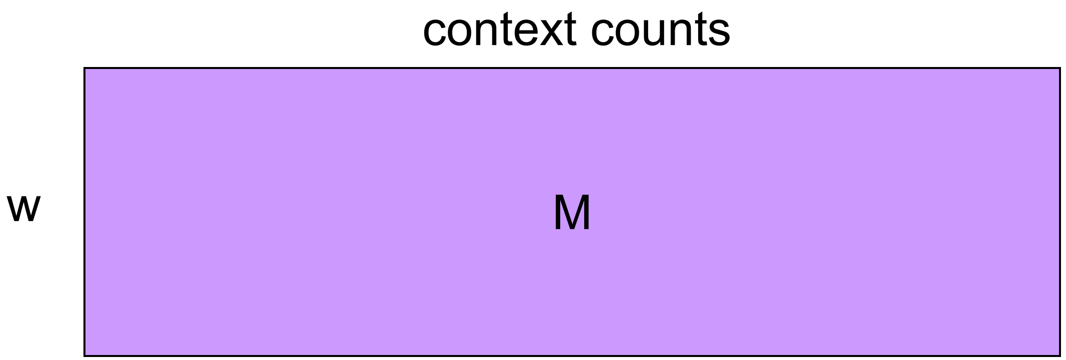
Dendrograms



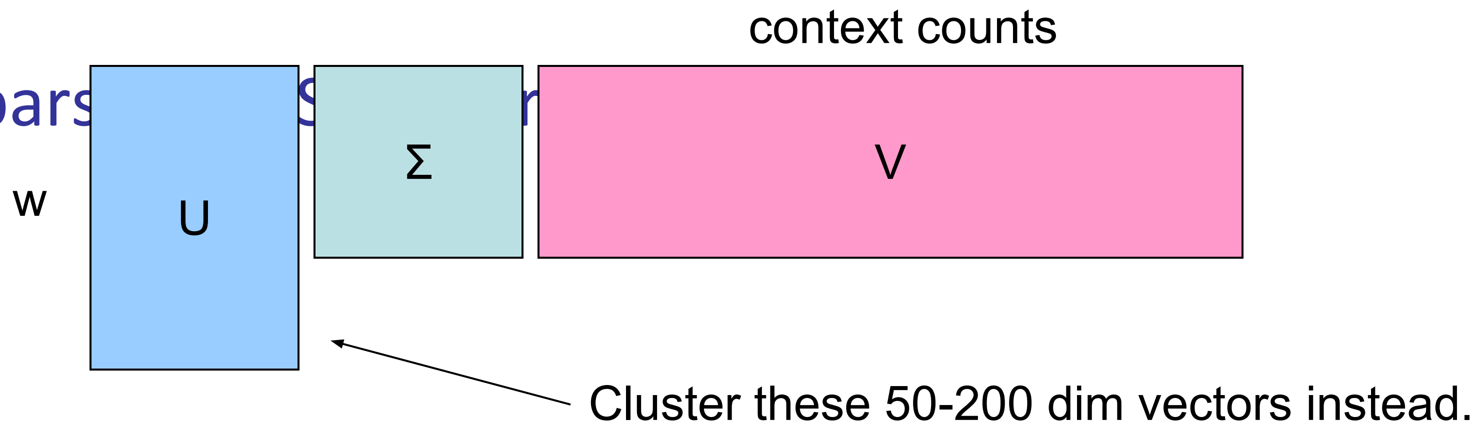


Vector Space Version

- [Shuetze 93] clusters words as points in R^n

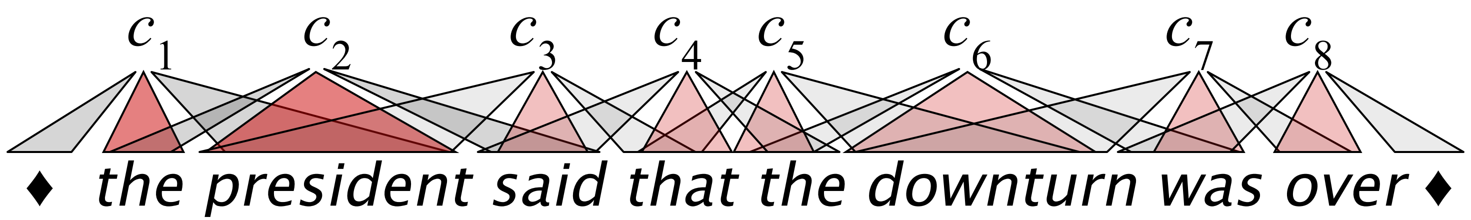


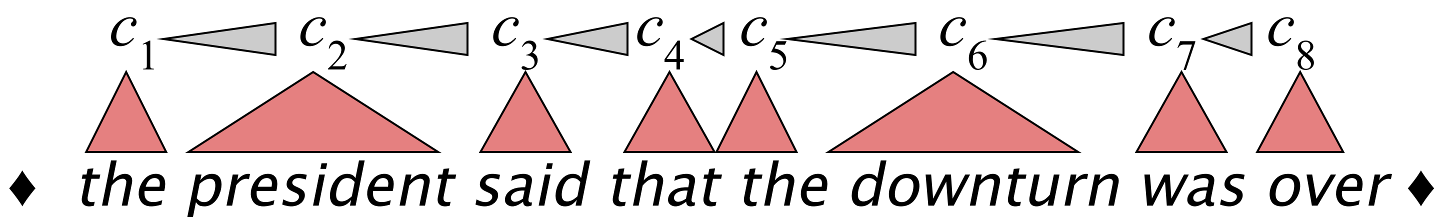
- Vectors too spars



A Probabilistic Version?

$$P(S,C) = \prod_i P(c_i)P(w_i \mid c_i)P(w_{i-1}, w_{i+1} \mid c_i)$$





What Else?

- Various newer ideas:
  - Context distributional clustering [Clark 00]
  - Morphology-driven models [Clark 03]
  - Contrastive estimation [Smith and Eisner 05]
  - Feature-rich induction [Haghighi and Klein 06]

- Also:
  - What about ambiguous words?
  - Using wider context signatures has been used for learning synonyms (what's wrong with this approach?)
  - Can extend these ideas for grammar induction (later)