Deep Learning for Efficient Discriminative Parsing

Ronan Collobert

ronan@collobert.com

Idiap Research Institute [†]

[†] Most of this work has been achieved at NEC Laboratories America

Motivation

- We have a generic architecture (see arXiv:1103.0398)
  for various Natural Language Processing tasks
  $\star$ part-of-speech $\star$ chunking $\star$ name entity recognition $\star$ semantic role

$$\text{The}|_{DT}\ \text{cat}|_{NN}\ \text{sat}|_{VBD}\ \text{on}_{IN}\ \text{the}|_{DT}\ \text{mat}|_{NN}$$

- Generic = "deep learning"

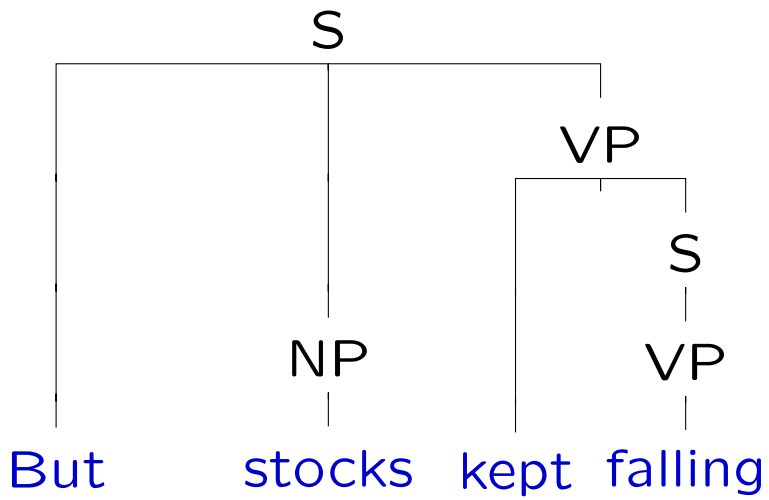
  trade "task-specific features" for "*learning the right features*"

- Leverage word representations trained on unlabeled corpus (852M words)

france	jesus	xbox	reddish	scratched	megabits
austria	god	amiga	greenish	nailed	octets
belgium	sati	playstation	bluish	smashed	mb/s
germany	christ	msx	pinkish	punched	bit/s
italy	satan	ipod	purplish	popped	baud
greece	kali	sega	brownish	crimped	carats
sweden	indra	psNUMBER	greyish	scraped	kbit/s
norway	vishnu	hd	grayish	screwed	megahertz
europe	ananda	dreamcast	whitish	sectioned	megapixels
hungary	parvati	geforce	silvery	slashed	gbit/s
switzerland	grace	capcom	yellowish	ripped	amperes

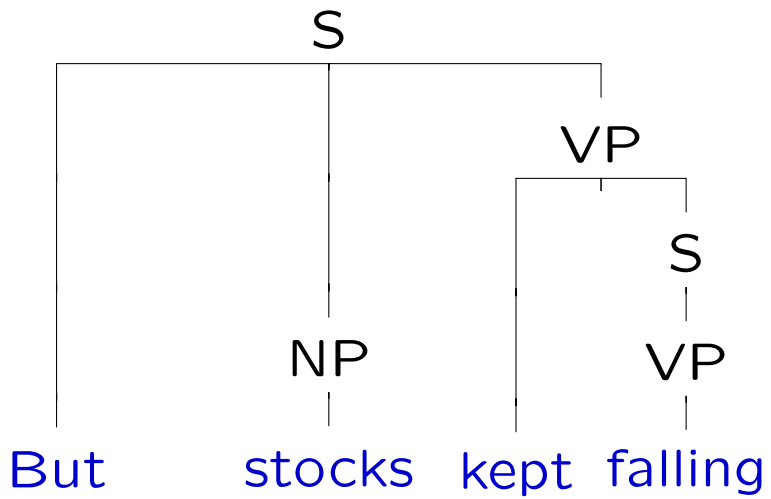
Motivation

- Can we do the same on syntactic parsing?
  (VP = Verb Phrase, NP = Noun Phrase, ...)



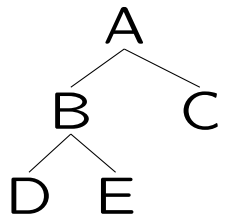
- Not another "flat" tagging task...
  - ⋆ Keep the same architecture
  - ⋆ Make it "recurrent"
  - ⋆ Add tree constraints

Standard Parsing Benchmarks



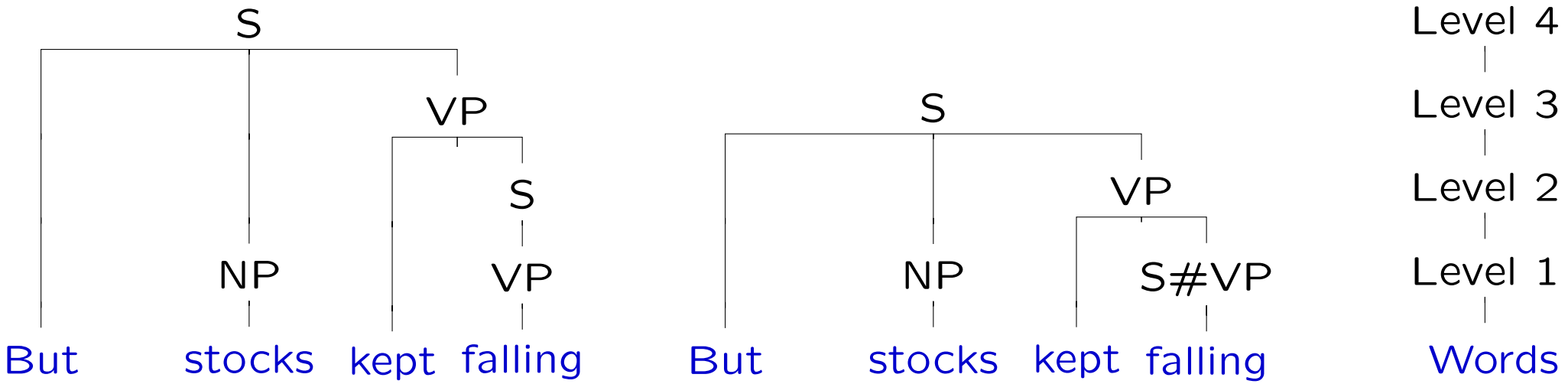
PCFG

$A \longrightarrow B\ C$



(Collins, 1999) (Charniak, 2000)	Lexicalized Probabilistic Context-Free Grammar (PCFG), POS, head words, chart parser, deleted interpolation, ... 30 pages of details in (Bikel, 2004)
(Charniak & Johnson, 2005 & 2006)	Re-ranking over the above, using lots of ad-hoc features
(Finkel et al, 2008) (Petrov & Klein, 2008) (Carreras & al, 2008)	PCFG, dependency features CRF or similar

Parsing as a Tagging Task



- BIOES tagging:

Level 3	B-S	I-S	I-S	E-S
Level 2	O	O	B-VP	E-VP
Level 1	O	S-NP	O	S-S#VP
Words	But	stocks	kept	falling

- **Greedy Approach:**

1. Predict Level 1.
2. Predict next level according to level history, with the same tagger†.
3. Update the history of levels and go to 2.

† Apply constraints during tagging to avoid loops and respect tree structure (Nodes strictly grow with tree levels)

Neural Networks

Stack several layers together (increase level of abstraction)
Parameters W^i trained by gradient descent

x	input (vector)
$W^1 \times \cdot$	linear operation (embedding)
$\tanh(\cdot)$	non-linearity
$W^2 \times \cdot$	linear operation (embedding)
	score per label

Convolutional layer

$X = (X_{\bullet 1}, X_{\bullet 2} \cdots)$	input (matrix)
$W \times \begin{pmatrix} X_{\bullet 1} & X_{\bullet 2} \\ X_{\bullet 2} & X_{\bullet 3} \cdots \\ X_{\bullet 3} & X_{\bullet 4} \end{pmatrix}$	convolution (local embedding for each input column)

Max over time layer

$X = (X_{\bullet 1}, X_{\bullet 2} \cdots)$	input (matrix)
$\max_t [X]_{i,t} \quad \forall i$	remove the time "dimension"

Words into Vectors

See (Bengio et al, 2001)

a word = index in a dictionary

The cat sat on the mat = $(w_1, w_2, w_3, w_4, w_5, w_6)$

binary code $\sim$ dictionary size

$$w \longleftrightarrow \left(0, \cdots 0, \underset{\text{at index } w}{1}, 0, \cdots 0\right)^{\mathrm{T}}$$

word embedding

$W \sim$ word representation size $\times$ dictionary size

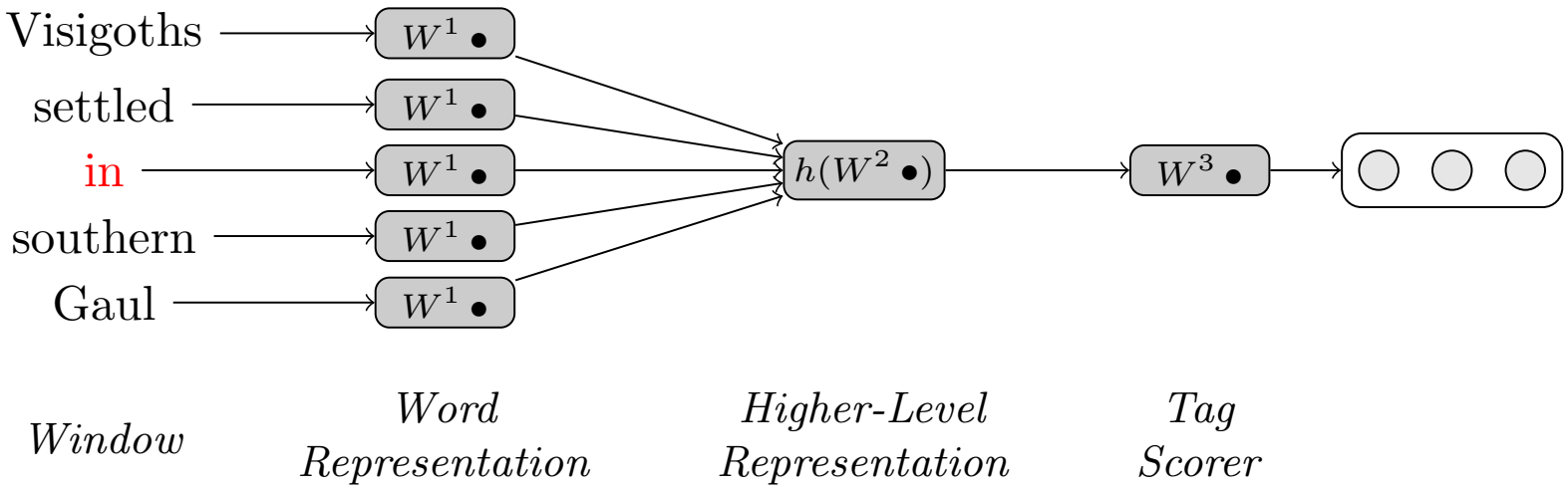
$$W \times \left(0, \cdots 0, \underset{\text{at index } w}{1}, 0, \cdots 0\right)^{\mathrm{T}} = W_{\bullet w}$$

lookup-table operation

Applicable to any discrete feature (words, caps, stems...)

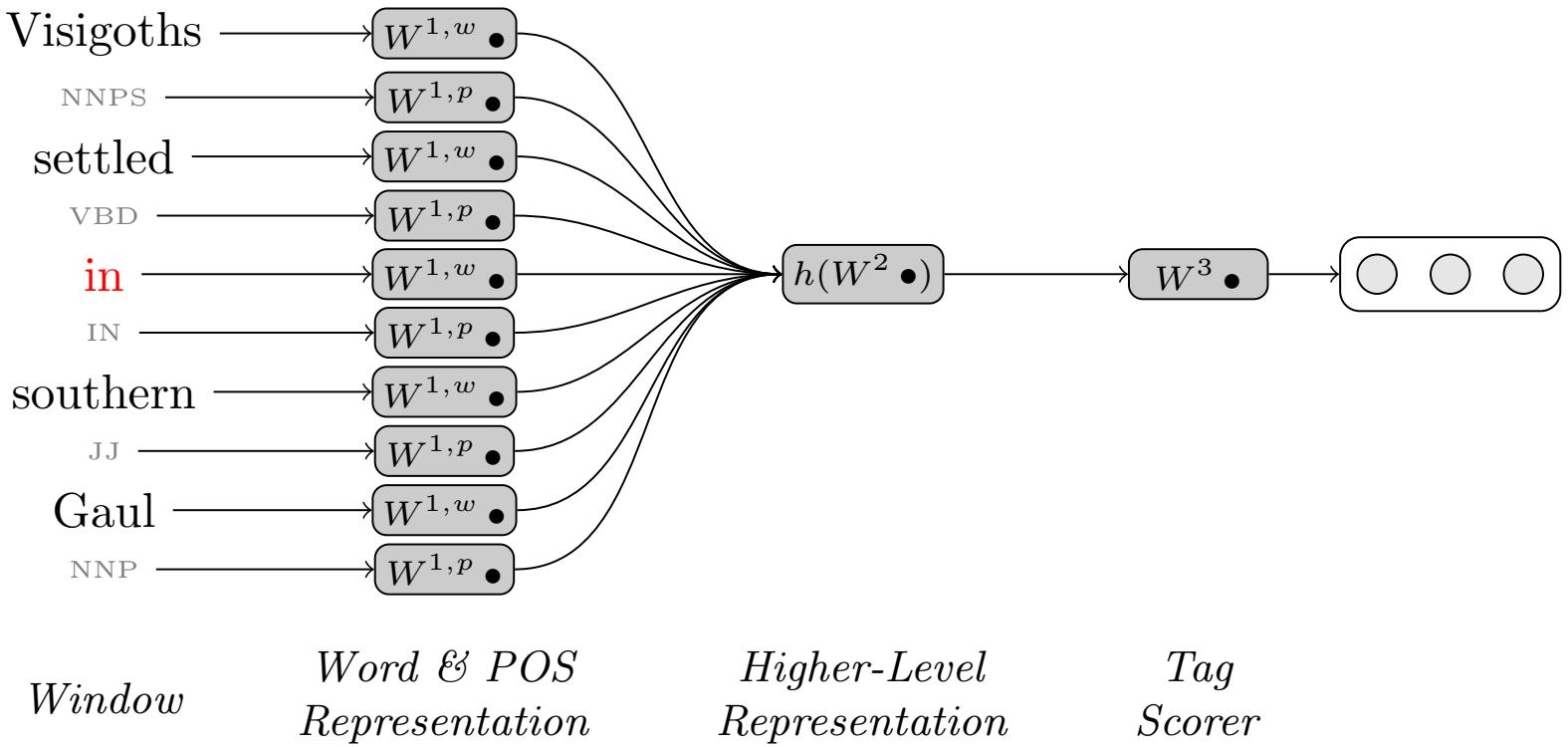
Window Approach

How to tag "in" in the sentence
"The Visigoths settled in southern Gaul"?



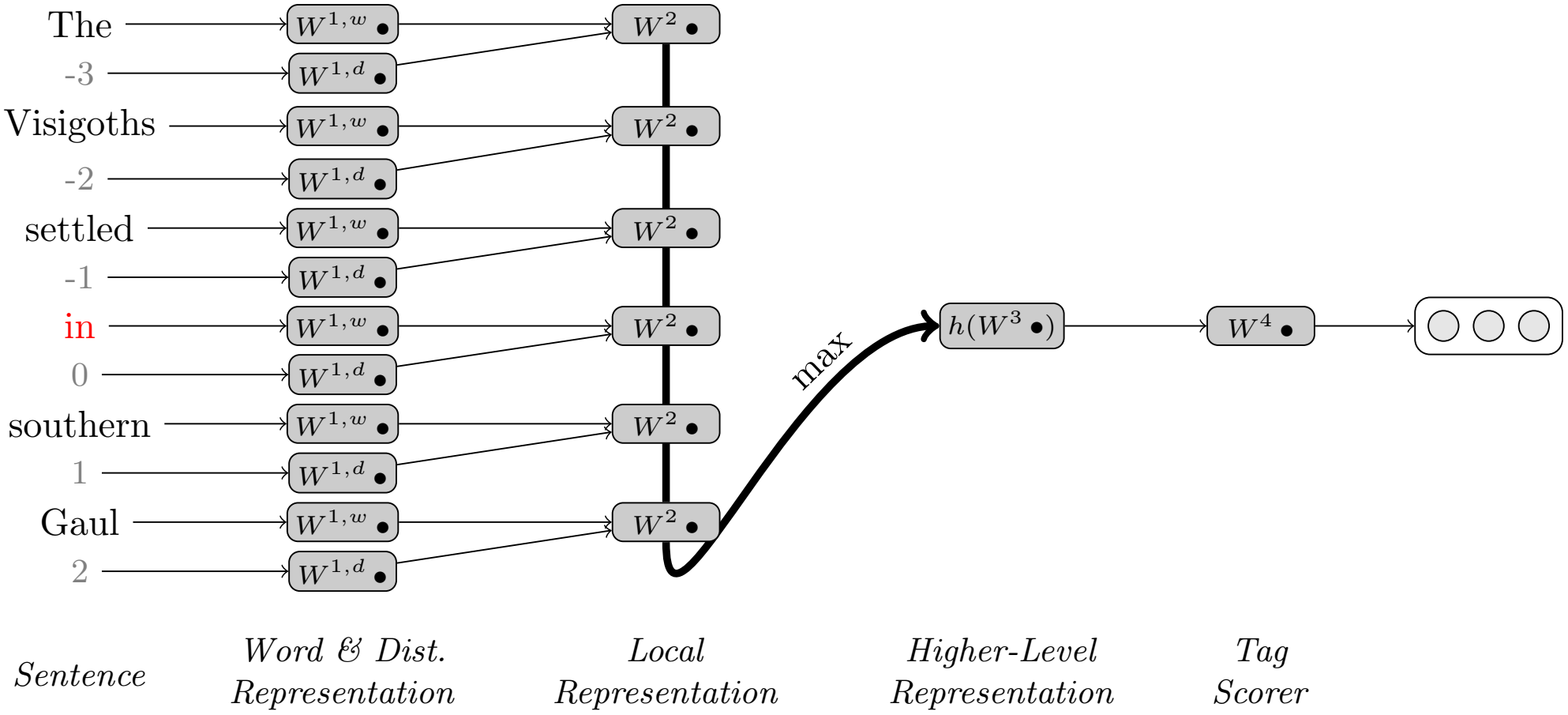
Window Approach (extra features)

How to tag "in" in the sentence
"The Visigoths settled in southern Gaul"?



Sentence Approach

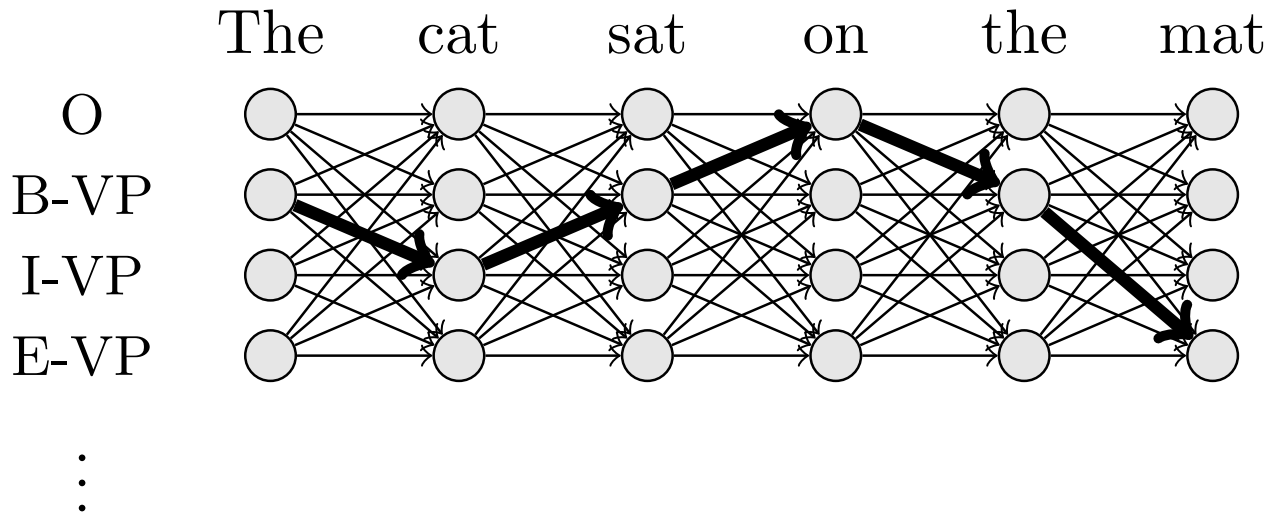
How to tag "in" in the sentence
"The Visigoths settled in southern Gaul"?



Sentence Scoring

- Sentence of T words $[\boldsymbol{w}]_1^T$
- Network parameters $\boldsymbol{\theta} = \{W^1, W^2, \dots\}$

- Tags are interdependent $\longrightarrow$ structured output learning
- Network score for each word w_t and tag k $\longrightarrow$ $f([\boldsymbol{w}]_1^T, k, t, \boldsymbol{\theta})$
- Transition score to jump from tag k to tag l $\longrightarrow$ A_{kl}



- Sentence score for a tag path $[i]_1^T$

$$s([\boldsymbol{w}]_1^T, [i]_1^T, \boldsymbol{\theta}) = \sum_{t=1}^{T} \Big(A_{[i]_{t-1}[i]_t} + f([\boldsymbol{w}]_1^T, [i]_t, t, \boldsymbol{\theta}) \Big)$$

- Inference: Viterbi algorithm

Training

- Conditional likelihood by normalizing w.r.t all possible paths:

$$\log p([y]_1^T \mid [\boldsymbol{w}]_1^T, \boldsymbol{\theta}) = s([\boldsymbol{w}]_1^T, [y]_1^T, \boldsymbol{\theta}) - \log \left[\sum_{\forall [j]_1^T} e^{s([\boldsymbol{w}]_1^T, [j]_1^T, \boldsymbol{\theta})} \right]$$

- Log-sum efficiently computed with recursive Forward algorithm
- Maximize likelihood over training set
- Use stochastic gradient ascent

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \lambda \frac{\partial \log p([y]_t^T \mid [\boldsymbol{x}]_1^T, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

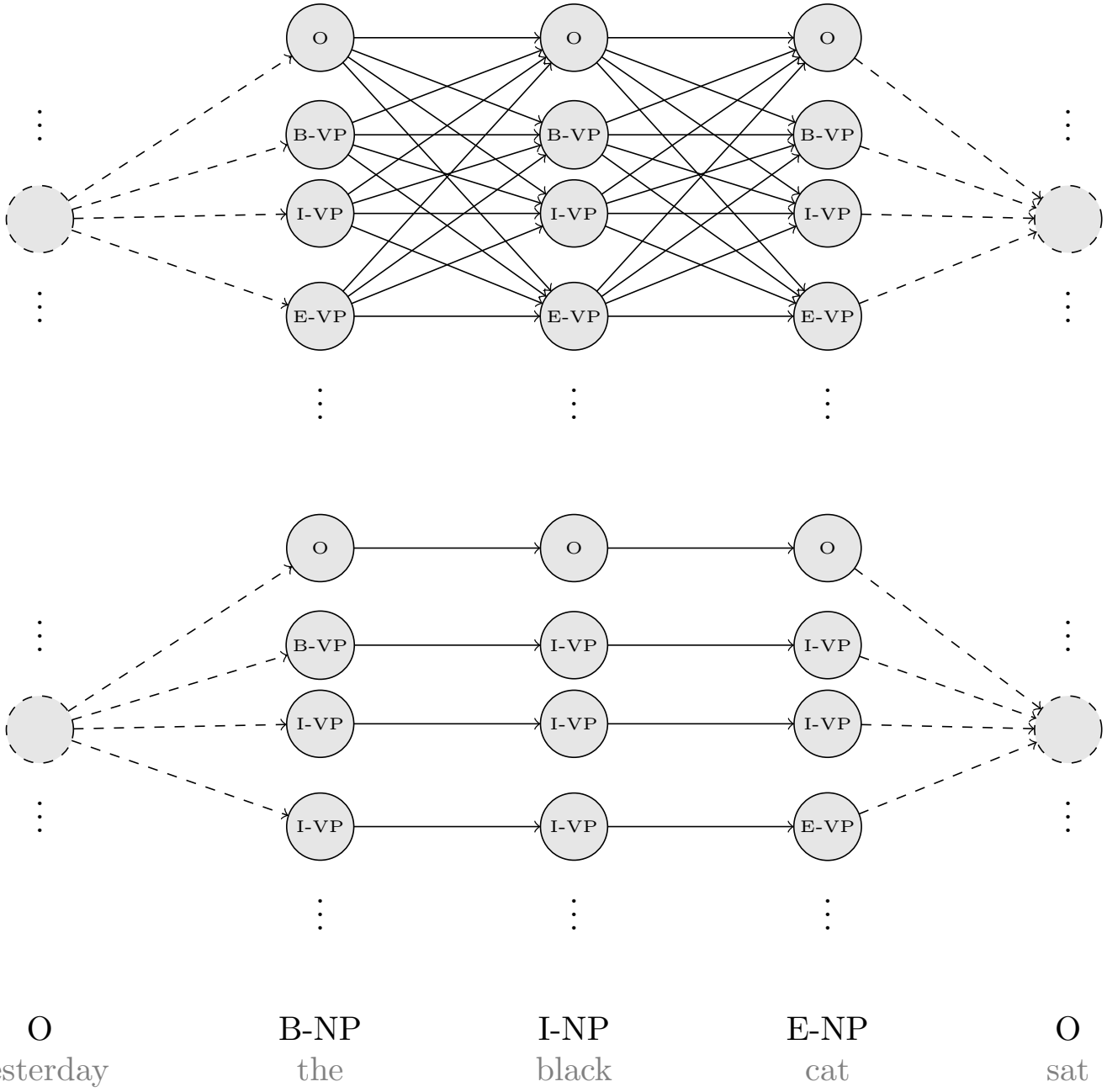
Fixed learning rate. "Tricks":

- ⋆ Divide learning rate by "fan-in"
- ⋆ Initialization according to "fan-in"

- Chain rule ("back-propagation") through Forward recursion and network

- Non-linear CRFs: Graph Transformer Networks (Bottou et al, 1997)

Constrained Graph: Leverage BIOES



Comparison With Pure Discriminative Parsers

- Recall, Precision and F1 scores on Penn Treebank, sentences ≤ 15 words
- Standard PARSEVAL evaluation

- Window approach (width=5) network
- Features:
  - ⋆ Raw low caps words (130K) + caps feature
  - ⋆ History: largest previously predicted chunks
  - ⋆ Our own POS (if mentioned)

- LM: word representations initialized with Language Model

Model	R	P	$F1$
Collins (1999)	88.2	89.2	88.7
Taskar et al. (2004)	89.1	89.1	89.1
Turian and Melamed (2006)	89.3	89.6	89.4
NN	82.4	82.8	82.6
NN+LM	86.1	87.2	86.6
NN+POS	87.1	86.2	86.7
NN+LM+POS	89.2	89.0	89.1

Comparison With State-Of-The-Art Parsers

- Recall, Precision and F1 scores on Penn Treebank
- Standard PARSEVAL evaluation
- Window & sentence approach networks
- Comparison with:
  - ⋆ "Benchmark" lexicalized parsers: Collins (1999), Charniak (2000)
  - ⋆ Re-ranking + self-training: McClosky, Charniak & Johnson (2006)
  - ⋆ Recent PCFG & CRF-based parsers

	≤ 40 Words			≤ 100 Words			Test Time
	R	P	F_1	R	P	F_1	(sec.)
Magerman (1995)	84.6	84.9	84.8				
Collins (1999)	88.5	88.7	88.6	88.1	88.3	88.2	2640
Charniak (2000)	90.1	90.1	90.1	89.6	89.5	89.6	1020
McClosky et al. (2006)						92.1	
Finkel et al. (2008)	88.8	89.2	89.0	87.8	88.2	88.0	
Petrov et al. (2008)			90.0			89.4	
Carreras et al. (2008)				89.9	91.1	90.5	
NN (window)	81.3	81.9	81.6	80.3	81.0	80.6	
NN+LM (window)	84.2	85.7	84.9	83.5	85.1	84.3	
NN+LM+POS (window)	85.6	86.8	86.2	84.8	86.2	85.5	
NN+LM+POS (sentence)	88.1	88.8	88.5	87.5	88.3	87.9	76

SENNA Demo

- Standalone, less than 3000 lines of C code
- Part-of-speech tagging, Chunking, Name entity recognition, Semantic Role Labeling, Parsing
- Available at `http://ml.nec-labs.com/software/senna`