

# Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che (HIT)

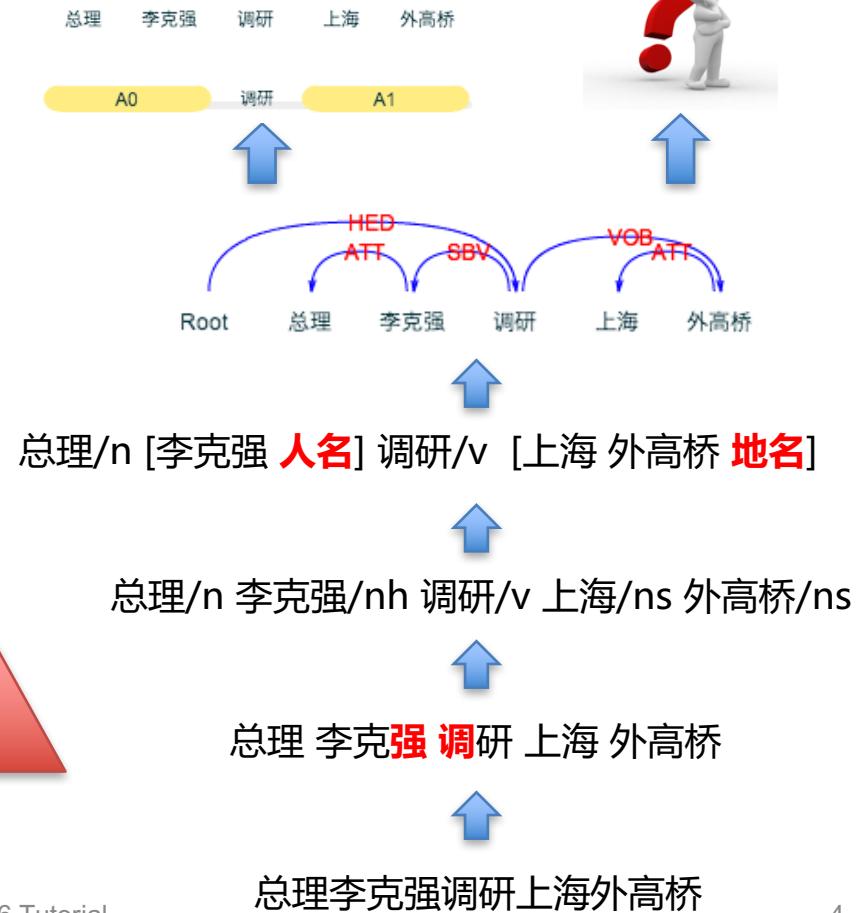
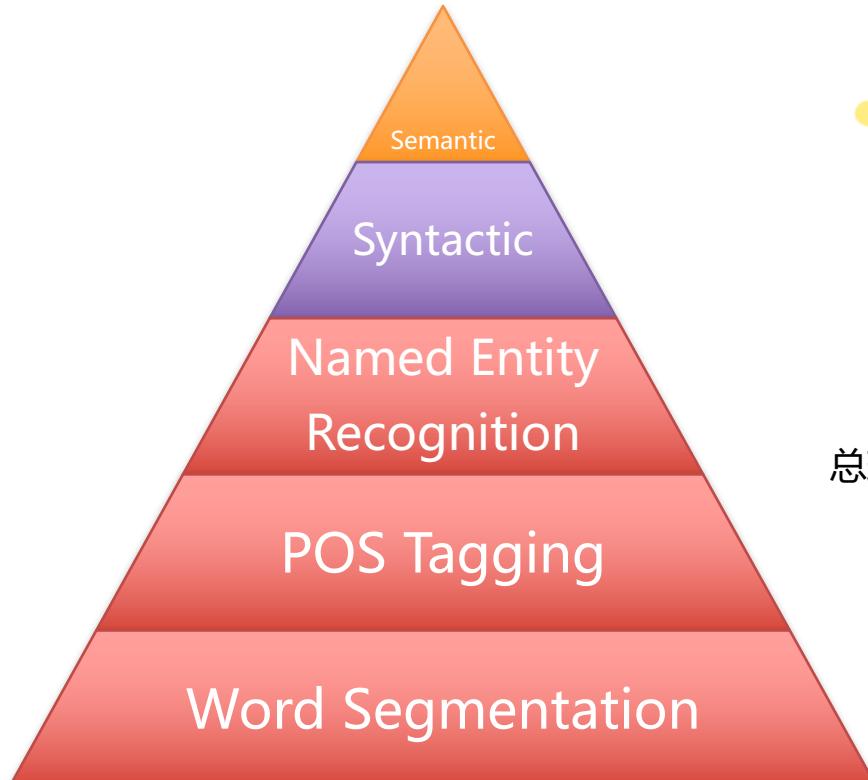
Yue Zhang (SUTD)

# Outline

Timeline	Content	Speaker
09:00-09:30	1. Introduction to Tasks	Wanxiang Che
09:30-10:00	2. Deep Learning Background	Wanxiang Che
10:00-10:10	Break	
10:10-10:40	3. Greedy Decoding	Yue Zhang
10:40-11:20	4. Dynamic Programming Decoding	Wanxiang Che
11:20-12:00	5. Beam-search Decoding	Yue Zhang

# Part 1: Introduction to Lexical, Syntactic and Semantic Analysis

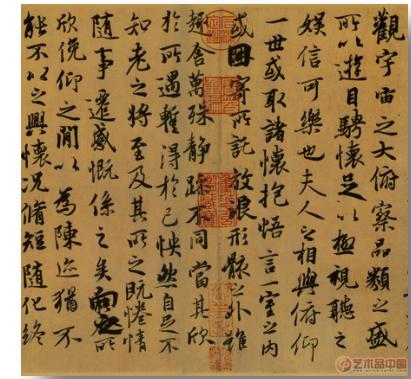
# NLP Pipeline



# Part 1.1: Background

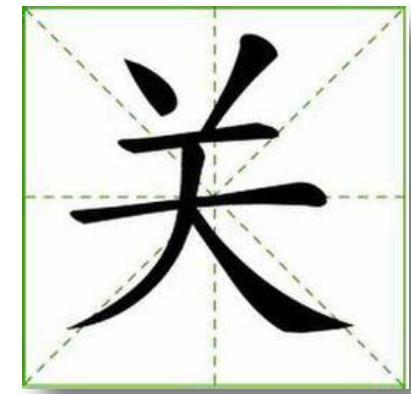
# Word Segmentation

- Words are fundamental semantic units
- Chinese has no obvious word boundaries
- Word segmentation
  - Split Chinese character sequence into words
- Ambiguities in word segmentation
  - E.g. 严守一 把手机关了
    - 严守一/ 把/ 手机/ 关/ 了
    - 严守/ 一把手/ 机关/ 了
    - 严守/ 一把/ 手机/ 关/ 了
    - 严守一/ 把手/ 机关/ 了
    - .....



# Part-of-speech (POS) Tagging

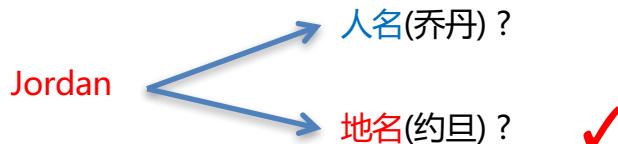
- A POS is a category of words which have similar grammatical properties
  - E.g. noun, verb, adjective
- POS tagging
  - Marking up a word in a text as a particular POS
  - based on both its definition and its context
- Ambiguities in POS Tagging
  - Time **flies** like an arrow.
  - 制服了敌人 vs. 穿着制服



# Named Entity Recognition (NER)

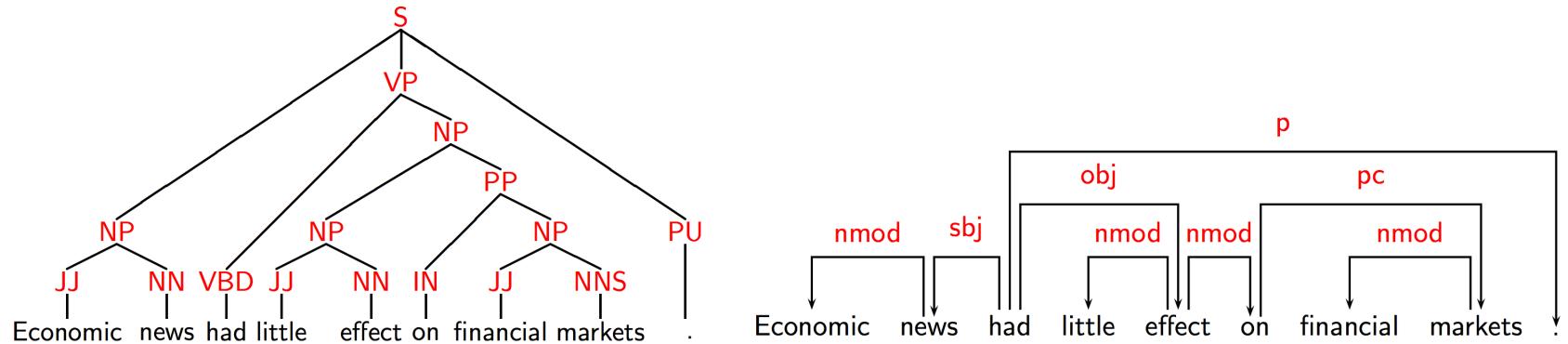
- Named Entities
  - Persons, locations, organizations, expressions of times, quantities, monetary values, percentages, etc
- Locating and classifying named entities in text into pre-defined categories
- Ambiguities in NER

Kerry to visit **Jordan**, Israel  
Palestinian peace on agenda.



# Syntactic Parsing

- Analyzing a natural language string conforming to the rules of a formal grammar, emphasizing subject, predicate, object, etc.
  - Constituency and Dependency Parsing



# Semantic Role Labeling

- Recognizing predicates and corresponding arguments
  - Yesterday <sub>time</sub>, Mary <sub>buyer</sub> bought a shirt <sub>bought thing</sub> from Tom <sub>seller</sub>
  - Whom <sub>buyer</sub> did Tom <sub>seller</sub> sell a shirt <sub>bought thing</sub> to, yesterday <sub>time</sub>
- Answer “Who did what to whom when and where”
  - Question Answering
  - Information Extraction
  - .....

# Combinatory Categorial Grammars (CCG)

$$\begin{array}{c} \text{CCG} \\ \hline NP \quad \text{is} \quad \text{fun} \\ \hline CCG \quad \frac{S \setminus NP / ADJ}{\lambda f. \lambda x. f(x)} \quad \frac{ADJ}{\lambda x. fun(x)} \\ \hline \frac{S \setminus NP}{\lambda x. fun(x)} \\ \hline \frac{S}{fun(CCG)} \end{array}$$

- CCG Lexical Entries
  - Pair words and phrases with meaning by a CCG category
- CCG Categories
  - Basic building block
  - Capture syntactic and semantic information jointly

Syntax       $ADJ : \lambda x. fun(x)$       Semantics

# Structured Prediction

- Predicting structured objects, rather than scalar discrete or real values
- Outputs are influenced each other
- For example
  - Sequence labeling/tagging
    - Given an input sequence, produce a label sequence of equal length.  
Each label is drawn from a small finite set
  - Parsing
    - Given an input sequence, build a tree whose structure obeys some grammar (compositional rules)

# Part 1.2: Sequence Labeling

# Sequence Labeling/Tagging

- Given an input sequence, produce a label sequence of equal length
- Each label is drawn from a small finite set
- Labels are influenced each other
- For example: POS tagging
  - Input
    - Profits soared at Boeing Co., easily topping forecasts on Wall Street, ...
  - Output
    - Profits/**N** soared/**V** at/**P** Boeing/**N** Co./**N** ,/, easily/**ADV** ...

# NER

- Input
  - Profits soared at Boeing Co., easily topping forecasts on Wall Street, ...
- Output
  - Profits soared at [Boeing Co. **ORG**], easily topping forecasts on [**Wall Street LOC**], ...
- Alternative Output (Tagging)
  - Profits/**O** soared/**O** at/**O** Boeing/**B-ORG** Co./**I-ORG** ,/**O** easily/**O** topping/**O** forecasts/**O** on/**O** Wall/**B-LOC** Street/**I-LOC** ,/**O** ...
- Where
  - B: Begin of entity XXX; I: Inside of entity XXX; O: Others

# Wore Segmentation

- Input
  - 严守一把手机关了
- Output
  - 严/守/一/把/手/机/关/了/
- Alternative Output (Tagging)
  - 严/B 守/I 一/I 把/B 手/B 机/I 关/B 了/B
- Where
  - B: Begin of a word; I: Inside of a word

# Semantic Role Labeling

- Input
  - Yesterday, Mary bought a shirt from Tom
- Output
  - [Yesterday <sub>time</sub>], [Mary <sub>buyer</sub>] bought/pred [a shirt <sub>bought thing</sub>] from [Tom <sub>seller</sub>]
- Alternative Output (Tagging)
  - Yesterday/B-time ,/O Mary/B-buyer bought/pred a/B-bought thing shirt/I-bought thing from/O Tom/B-seller
- Where
  - B: Begin of an arg; I: Inside of an arg; O: Others

# CCG Supertagging

<i>He</i>	<i>goes</i>	<i>on</i>	<i>the</i>	<i>road</i>	<i>with</i>	<i>his</i>	<i>piano</i>
$\overline{NP}$	$(S[dcl]\backslash NP)/PP$	$\overline{PP/NP}$	$\overline{NP/N}$	$\overline{N}$	$\overline{((S\backslash NP)\backslash (S\backslash NP))/NP}$	$\overline{NP/N}$	$\overline{N}$
<i>A</i>	<i>bitter</i>	<i>conflict</i>	<i>with</i>	<i>global</i>	<i>implications</i>		
$\overline{NP/N}$	$\overline{N/N}$	$\overline{N}$	$\overline{(NP\backslash NP)/NP}$	$\overline{N/N}$	$\overline{N}$		

frequency cut-off	# cat types	# cat tokens in 2-21 not in cat set	# sentences in 2-21 with missing cat	# cat tokens in 00 not in cat set	# sentences in 00 with missing cat
1	1 225	0	0	12 (0.03%)	12 (0.6%)
10	409	1 933 (0.2%)	1 712 (4.3%)	79 (0.2%)	69 (3.6%)

# Sequence Labeling Models

HMM

$$P(y_{[1:n]}, x_{[1:n]}) \propto \prod_{t=1}^n P(y_t | y_{t-1}) P(x_t | y_t)$$

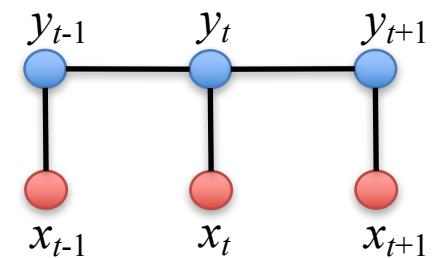
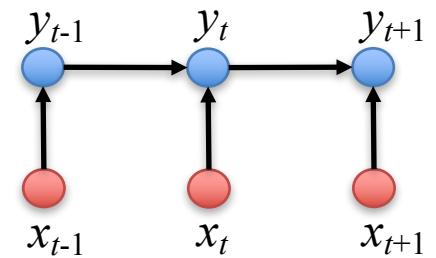
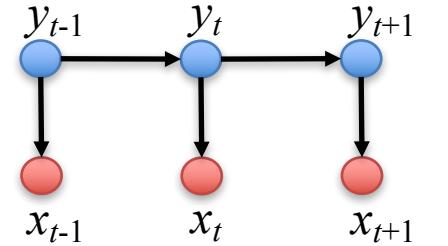
MEMM

$$P(y_{[1:n]} | x_{[1:n]}) \propto \prod_{t=1}^n P(y_t | y_{t-1}, x_t)$$

$$\propto \prod_{t=1}^n \frac{1}{Z_{y_{t-1}, x_t}} \exp \left( \begin{array}{l} \sum_j \lambda_j f_j(y_t, y_{t-1}) \\ + \sum_k \mu_k g_k(y_t, x_t) \end{array} \right)$$

CRF

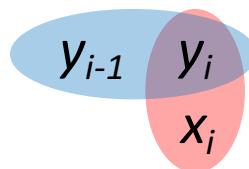
$$P(y_{[1:n]} | x_{[1:n]}) \propto \frac{1}{Z_{y_{[1:n]}}} \prod_{t=1}^n \exp \left( \begin{array}{l} \sum_j \lambda_j f_j(y_t, y_{t-1}) \\ + \sum_k \mu_k g_k(y_t, x_t) \end{array} \right)$$



# Features of POS Tagging with CRF

- Assume only two feature templates

- tag bigrams



- word/tag pairs

$$f_{100} = \begin{cases} 1 & \text{if } \langle y_{i-1}, y_i \rangle = \langle n, v \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$g_{101} = \begin{cases} 1 & \text{if } x_i \text{ is ended with "ing" and } y_i = v \\ 0 & \text{otherwise} \end{cases}$$

# CRF Decoding

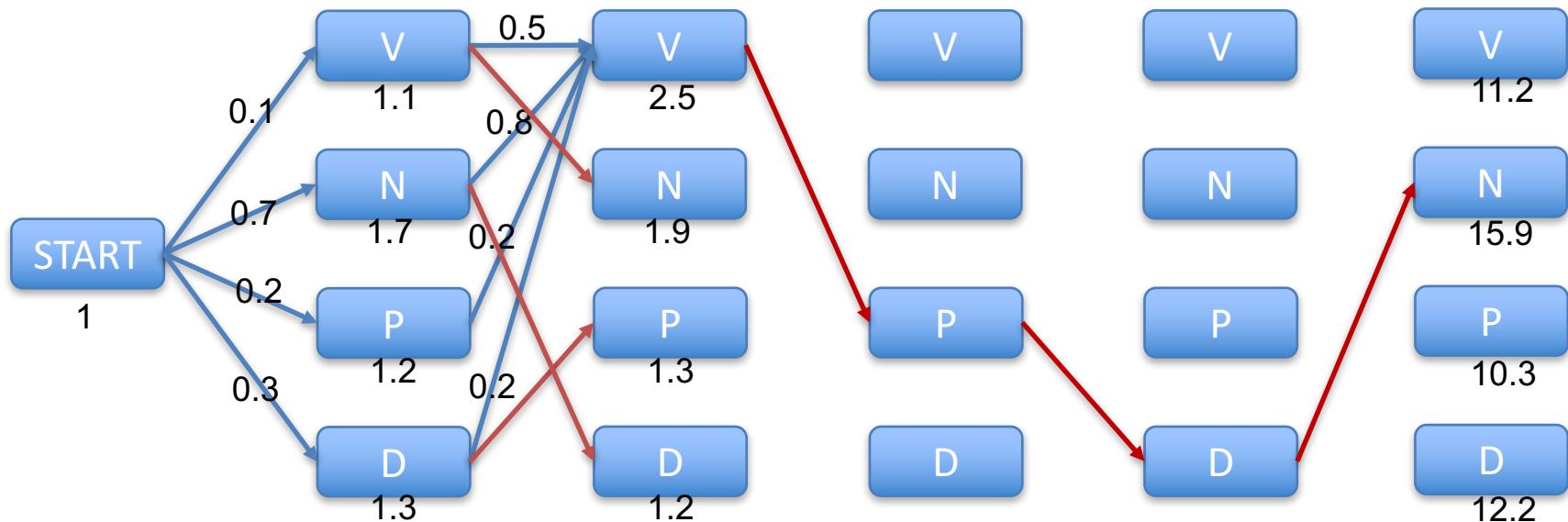
$$\arg \max_{y_{[1:n]} \in \text{GEN}(x_{[1:n]})} \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(x_{[1:n]}, y_i, y_{i-1})$$

where  $\text{GEN}(x_{[1:n]})$  is all possible tag sequences

- Dynamic Programming Algorithm
  - Viterbi Algorithm

# Viterbi Algorithm

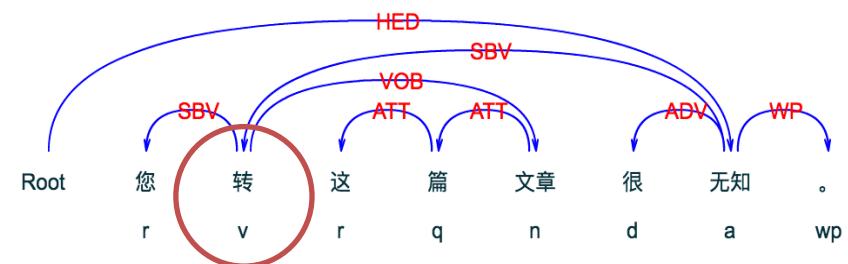
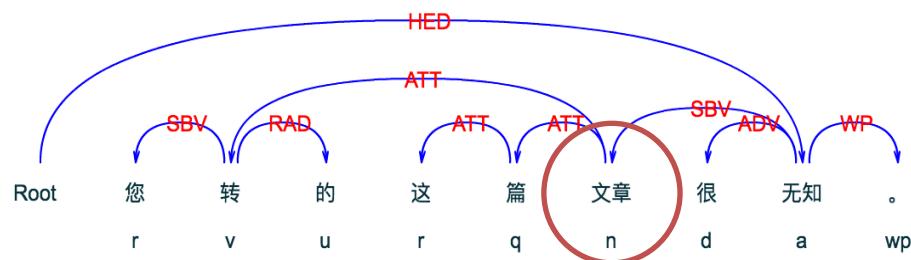
- Define a dynamic programming table
  - $\pi(i, y) = \text{maximum score of a tag sequence ending in tag } y \text{ at position } i$
- Recursive definition:  $\pi(i, y) = \max_t (\pi(i - 1, t) + \mathbf{w} \cdot \mathbf{f}(x_{[1:n]}, y, t))$



# Part 1.3: Parsing

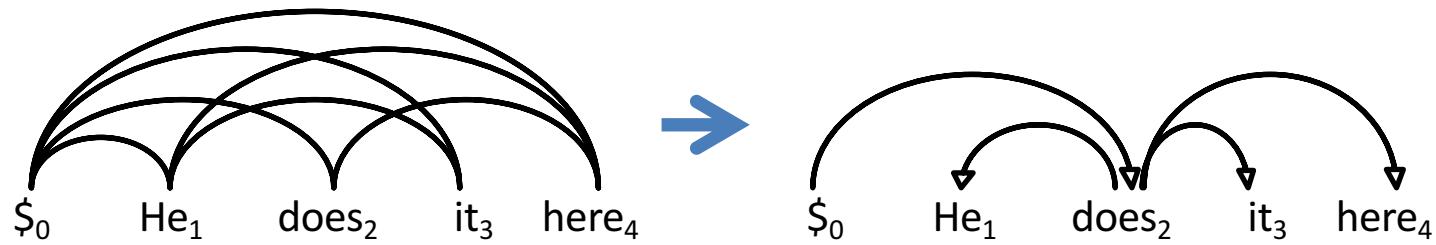
# Dependency Parsing

- A dependency tree is a tree structure composed of the input words and meets a few constraints:
  - Single-head
  - Connected
  - Acyclic



# Graph-based Dependency Parsing

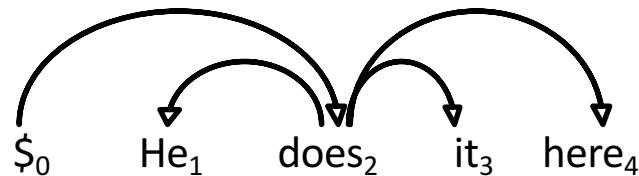
- Find the highest scoring tree from a complete dependency graph



$$Y^* = \arg \max_{Y \in \Phi(X)} score(X, Y)$$

# First-order as an Example

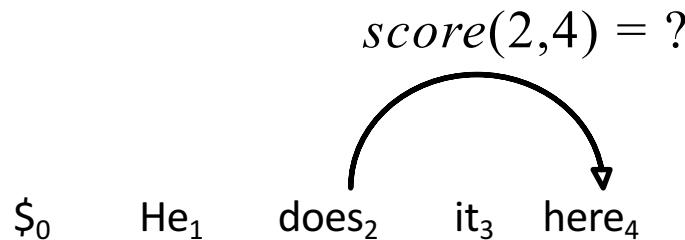
- The first-order graph-based method assumes that arcs in a tree are independent from each other (arc-factorization)
- Maximum Spanning Tree (MST) Algorithm



$$score(X, Y) = \sum_{(h,m) \in Y} score(X, h, m)$$

# How to Score an Arc

- Given a sentence, how to determine the score of each arc?



- Feature based representation: an arc is represented as a feature vector  $\mathbf{f}(2,4)$

$$score(2,4) = \mathbf{w} \cdot \mathbf{f}(2,4)$$

# Features for an Arc



*	As	McGwire	neared	,	fans	went	wild	
	[went]		[VBD]		[As]		[ADP]	[went]
	[VERB]		[As]		[IN]		[went, VBD]	[As, ADP]
	[went, As]		[VBD, ADP]		[went, VERB]		[As, IN]	[went, As]
	[VERB, IN]		[VBD, As, ADP]		[went, As, ADP]		[went, VBD, ADP]	[went, VBD, AS]
	[ADJ, *, ADP]		[VBD, *, ADP]		[VBD, ADJ, ADP]		[VBD, ADJ, *]	[NNS, *, ADP]
	[NNS, VBD, ADP]		[NNS, VBD, *]		[ADJ, ADP, NNP]		[VBD, ADP, NNP]	[VBD, ADJ, NNP]
	[NNS, ADP, NNP]		[NNS, VBD, NNP]		[went, left, 5]		[VBD, left, 5]	[As, left, 5]
	[ADP, left, 5]		[VERB, As, IN]		[went, As, IN]		[went, VERB, IN]	[went, VERB, As]
	[JJ, *, IN]		[VERB, *, IN]		[VERB, JJ, IN]		[VERB, JJ, *]	[NOUN, *, IN]
	[NOUN, VERB, IN]		[NOUN, VERB, *]		[JJ, IN, NOUN]		[VERB, IN, NOUN]	[VERB, JJ, NOUN]
	[NOUN, IN, NOUN]		[NOUN, VERB, NOUN]		[went, left, 5]		[VERB, left, 5]	[As, left, 5]
	[IN, left, 5]		[went, VBD, As, ADP]		[VBD, ADJ, *, ADP]		[NNS, VBD, *, ADP]	[VBD, ADJ, ADP, NNP]
	[NNS, VBD, ADP, NNP]		[went, VBD, left, 5]		[As, ADP, left, 5]		[went, As, left, 5]	[VBD, ADP, left, 5]
	[went, VERB, As, IN]		[VERB, JJ, *, IN]		[NOUN, VERB, *, IN]		[VERB, JJ, IN, NOUN]	[NOUN, VERB, IN, NOUN]
	[went, VERB, left, 5]		[As, IN, left, 5]		[went, As, left, 5]		[VERB, IN, left, 5]	[VBD, As, ADP, left, 5]
	[went, As, ADP, left, 5]		[went, VBD, ADP, left, 5]		[went, VBD, As, left, 5]		[ADJ, *, ADP, left, 5]	[VBD, *, ADP, left, 5]
	[VBD, ADJ, ADP, left, 5]		[VBD, ADJ, *, left, 5]		[NNS, *, ADP, left, 5]		[NNS, VBD, ADP, left, 5]	[NNS, VBD, *, left, 5]
	[ADJ, ADP, NNP, left, 5]		[VBD, ADP, NNP, left, 5]		[VBD, ADJ, NNP, left, 5]		[NNS, ADP, NNP, left, 5]	[NNS, VBD, NNP, left, 5]
	[VERB, As, IN, left, 5]		[went, As, IN, left, 5]		[went, VERB, IN, left, 5]		[went, VERB, As, left, 5]	[JJ, *, IN, left, 5]
2016-10-14	[VERB, *, IN, left, 5]		[VERB, JJ, IN, left, 5]		[VERB, QL, 20165Tutorial]		[NOUN, *, IN, left, 5]	[NOUN, VERB, IN, left, 5]

# Decoding for first-order model

- Eisner (2000) described a **dynamic programming** based decoding algorithm for bilexical grammar
- McDonald+ (2005) applied this algorithm to the search problem of the first-order model

# Transition-based Dependency Parsing

- Gradually build a tree by applying a sequence of transition actions – shift/reduce (Yamada and Matsumoto, 2003; Nivre, 2003)
- The score of the tree is equal to the summation of the scores of the actions

$$score(X, Y) = \sum_{i=0}^m score(X, h_i, a_i)$$

$a_i$  → the action adopted in step  $i$

$h_i$  → the partial results built so far by  $a_0 \dots a_{i-1}$

$Y$  → the tree built by the action sequence  $a_0 \dots a_m$

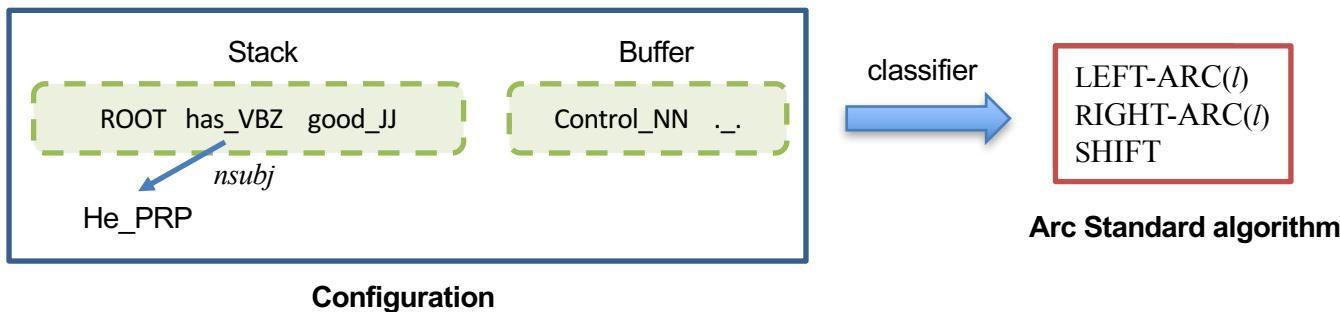
# Transition-based Dependency Parsing

- The goal of a transition-based dependency parser is to find the highest scoring action sequence that builds a legal tree.

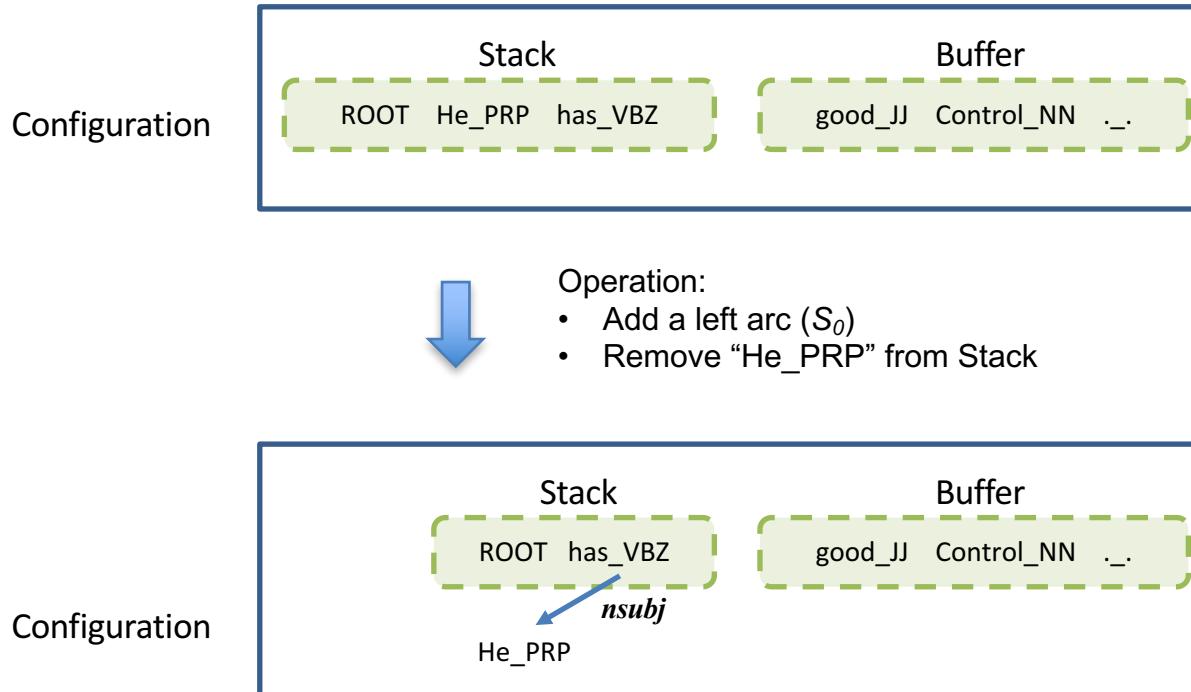
$$\begin{aligned} Y^* &= \arg \max_{Y \in \Phi(X)} \text{score}(X, Y) \\ &= \arg \max_{a_0 \dots a_m \rightarrow Y} \sum_{i=0}^m \text{score}(X, h_i, a_i) \end{aligned}$$

# Transition-based Dependency Parsing

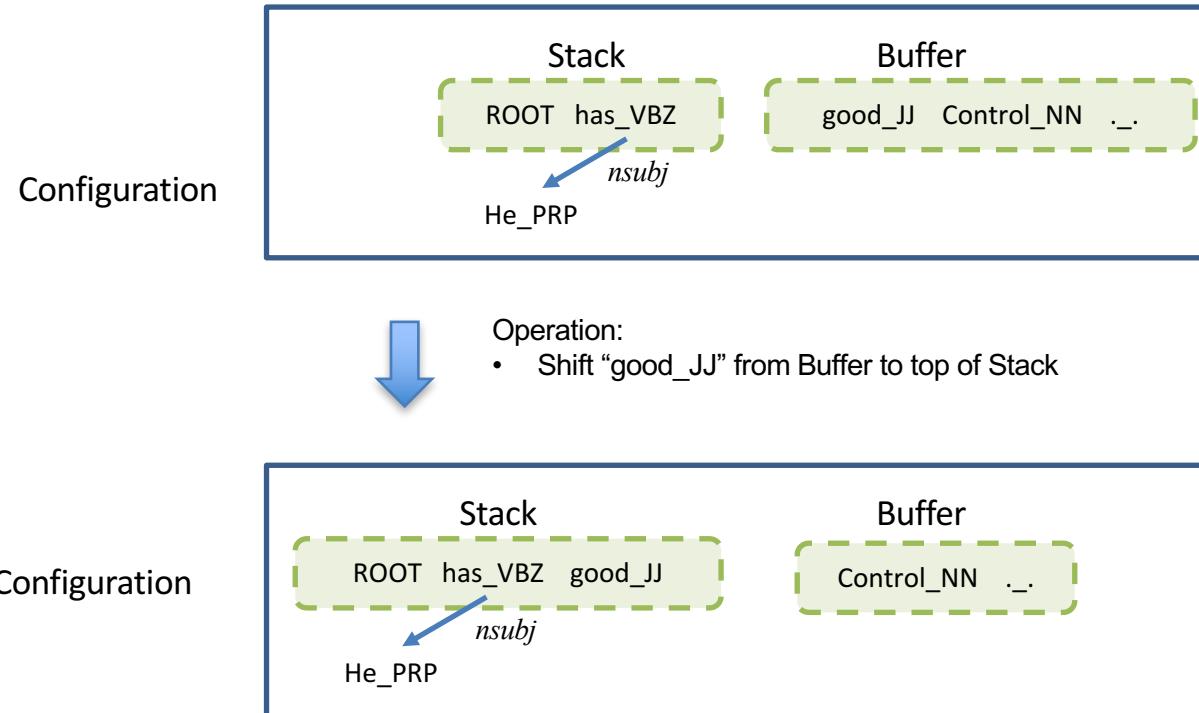
- Greedily predict a transition sequence from an initial parser state to some terminal states
- State (configuration)  
= Stack + Buffer + Dependency Arcs



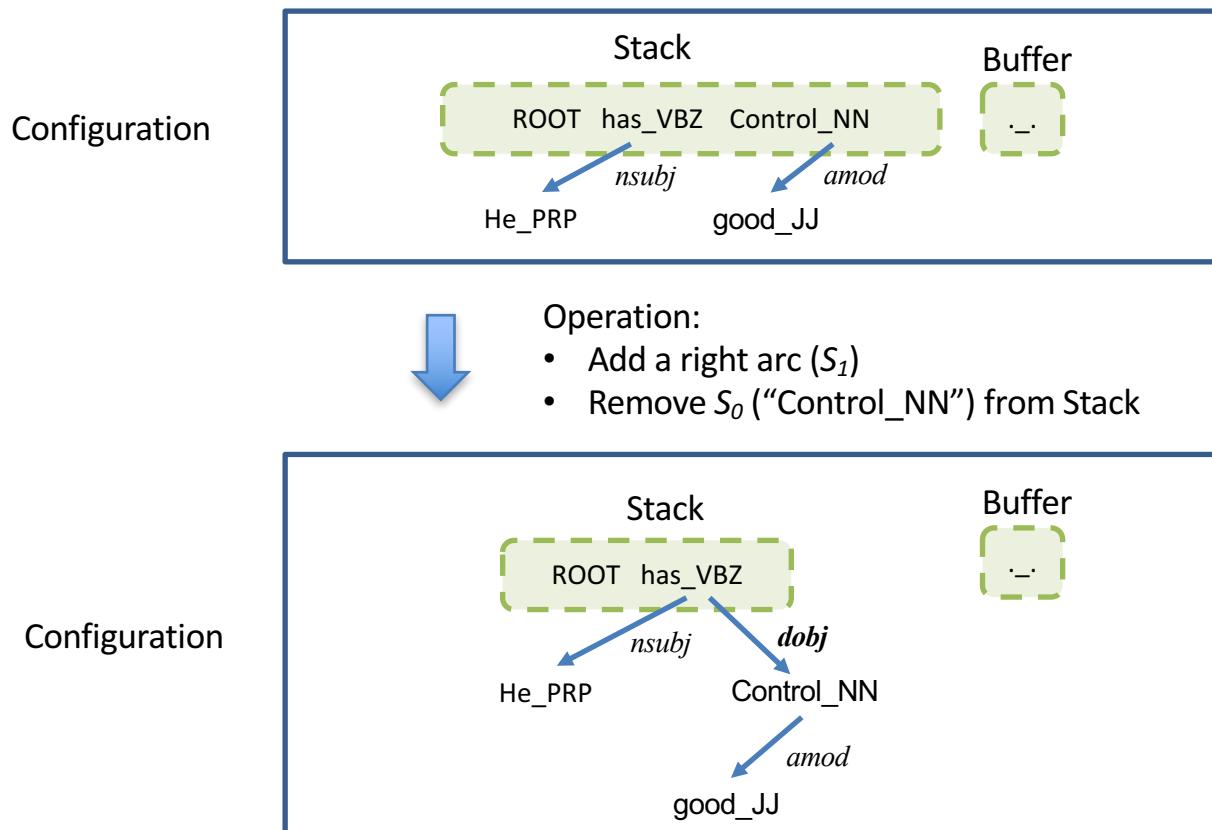
# Transition Action: LEFT-ARC (/)



# Transition Action: SHIFT



# Transition Action: RIGHT-ARC (/)



Stack

ROOT

has\_VBZ

He\_PRP  
*nsubj*

Control\_NN  
*dobj*

good\_JJ  
*amod*

Buffer

...  
...

# An Example

## Arc-standard Algorithm

### 初始状态

Stack只有根节点，待处理词在Buffer中

### SHIFT

将Buffer中第一个词压入Stack

### LEFT-ARC

弹出Stack中第二个词，生成一条弧从栈顶词指向第二个词

### RIGHT-ARC

弹出栈顶词，生成一条弧从栈顶第二个词指向栈顶词

### 终结状态

Stack只有根节点，Buffer为空

### Stack

ROOT

### Buffer

小明\_nr 吃\_v 苹果\_n

ROOT 小明\_nr

v

吃\_v 苹果\_n

ROOT 吃\_v 苹果\_n

sub

小明\_nr

ROOT

HED

sub

吃\_v

小明\_nr

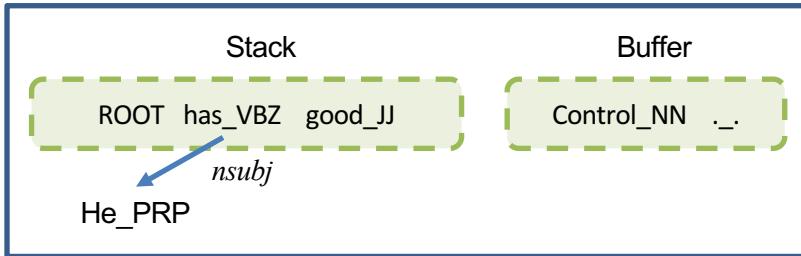
obj

苹果\_n

小明\_nr 苹果\_n

# Traditional Features

Configuration



Feature Vector:

- Binary
- Sparse
- High-dimensional



**Feature templates:** a combination of elements from the configuration.

- For example: (Zhang and Nivre, 2011): 72 feature templates

from single words

$S_0wp; S_0w; S_0p; N_0wp; N_0w; N_0p;$   
 $N_1wp; N_1w; N_1p; N_2wp; N_2w; N_2p;$

from word pairs

$S_0wpN_0wp; S_0wpN_0w; S_0wN_0wp; S_0wpN_0p;$   
 $S_0pN_0wp; S_0wN_0w; S_0pN_0p$   
 $N_0pN_1p$

from three words

$N_0pN_1pN_2p; S_0pN_0pN_1p; S_0hpS_0pN_0p;$   
 $S_0pS_0lpN_0p; S_0pS_0rpN_0p; S_0pN_0pN_0lp$

Table 1: Baseline feature templates.

$w$  – word;  $p$  – POS-tag.

distance

$S_0wd; S_0pd; N_0wd; N_0pd;$   
 $S_0wN_0wd; S_0pN_0pd;$

valency

$S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;$

unigrams

$S_0hw; S_0hp; S_0l; S_0tw; S_0up; S_0tl;$   
 $S_0rw; S_0rp; S_0rl; N_0lw; N_0lp; N_0ll;$

third-order

$S_{0h2}w; S_{0h2}p; S_{0h}l; S_{0l2}w; S_{0l2}p; S_{0l2}l;$   
 $S_{0r2}w; S_{0r2}p; S_{0r2}l; N_{0l2}w; N_{0l2}p; N_{0l2}l;$   
 $S_0pS_0lpS_{0l2}p; S_0pS_0rpS_{0r2}p;$   
 $S_0pS_0hpS_{0h2}p; N_0pN_0lpN_{0l2}p;$

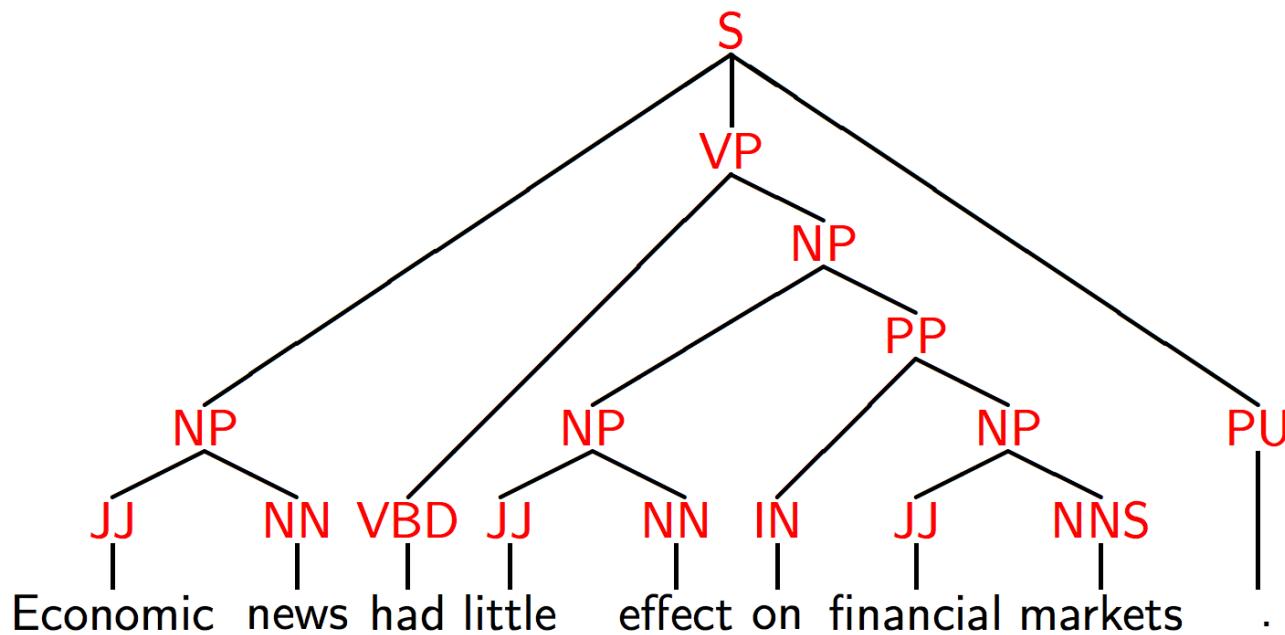
label set

$S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;$

Table 2: New feature templates.

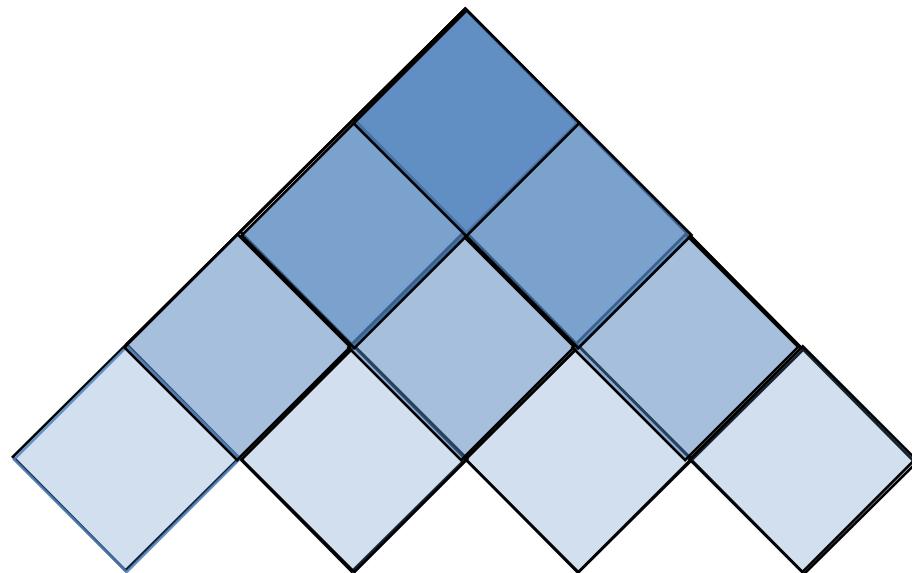
$w$  – word;  $p$  – POS-tag;  $v_l, v_r$  – valency;  $l$  – dependency label,  $s_l, s_r$  – labelset.<sup>37</sup>

# Constituency Parsing



# Constituency Parsing

- Chart-based
  - E.g. Cocke–Younger–Kasami algorithm (CYK or CKY)
  - A kind of Dynamic Programming



fish people fish tanks

## PCFG

Rule Prob  $\theta_i$ ,

$S \rightarrow NP VP$	$\theta_0$
$NP \rightarrow NP NP$	$\theta_1$
...	
$N \rightarrow \text{fish}$	$\theta_{42}$
$N \rightarrow \text{people}$	$\theta_{43}$
$V \rightarrow \text{fish}$	$\theta_{44}$

# CKY Parsing Algorithm

**Input:** a sentence  $s = x_1 \dots x_n$ , a PCFG  $G = (N, \Sigma, S, R, q)$ .

**Initialization:**

For all  $i \in \{1 \dots n\}$ , for all  $X \in N$ ,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

**Algorithm:**

- For  $l = 1 \dots (n - 1)$ 
  - For  $i = 1 \dots (n - l)$ 
    - \* Set  $j = i + l$
    - \* For all  $X \in N$ , calculate

$$\boxed{\pi(i, j, X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \dots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))}$$

and

$$bp(i, j, X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \dots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$

**Output:** Return  $\pi(1, n, S) = \max_{t \in T(s)} p(t)$ , and backpointers  $bp$  which allow recovery of  $\arg \max_{t \in T(s)} p(t)$ .

CCL 2016 Tutorial

# Summarization

- Classical NLP Methods

Problem		Model	Decoding	NLP Tasks
Sequence Labeling		CRF	Dynamic Programming	POS tagging, Word Segmentation, NER, SRL, CCG Supertagging
Parsing	Dependency	Transition-based	Greedy/Beam Search	Dependency Parsing
		Graph-based	Dynamic Programming	
	Constituency	Chart-based	Constituency Parsing	

Lots of feature engineering work!

# Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che and Yue Zhang

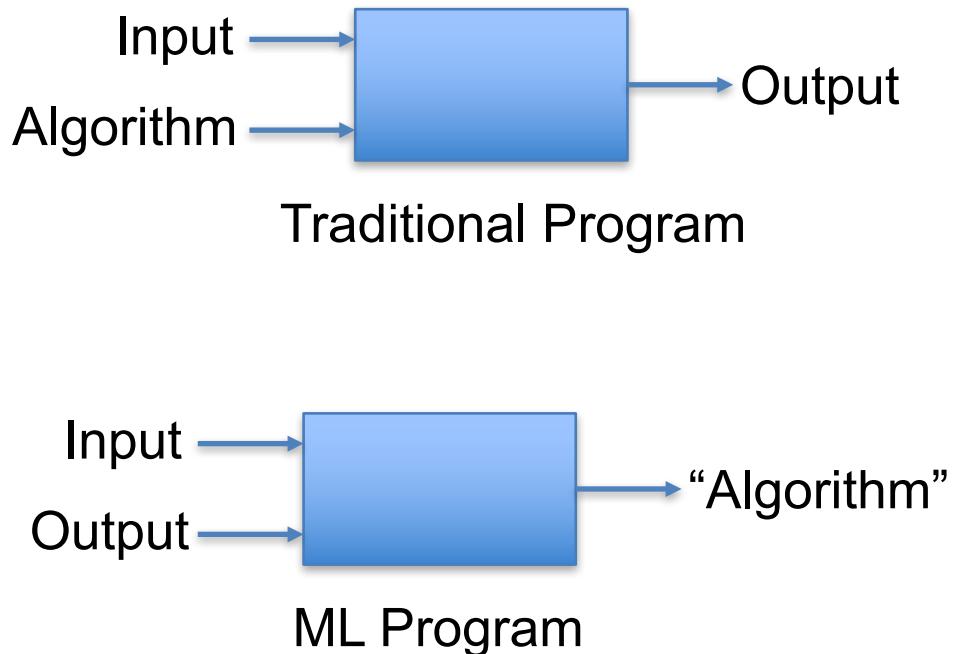
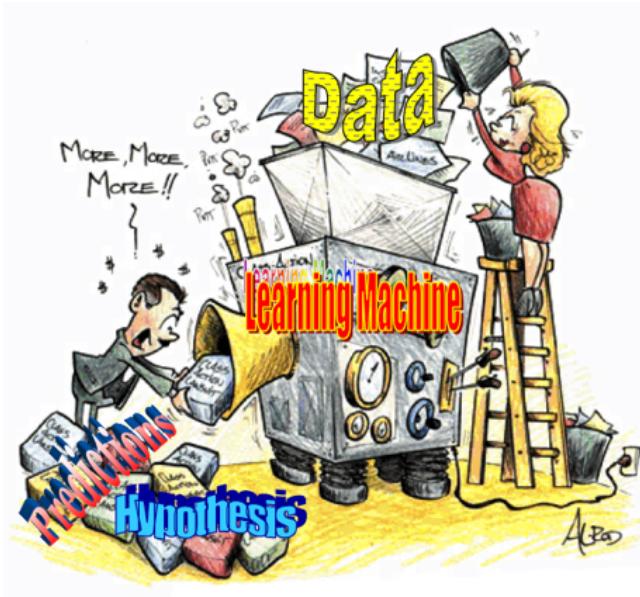
2016-10

# Part 2: Introduction to Deep Learning

# Part 2.1: Deep Learning Background

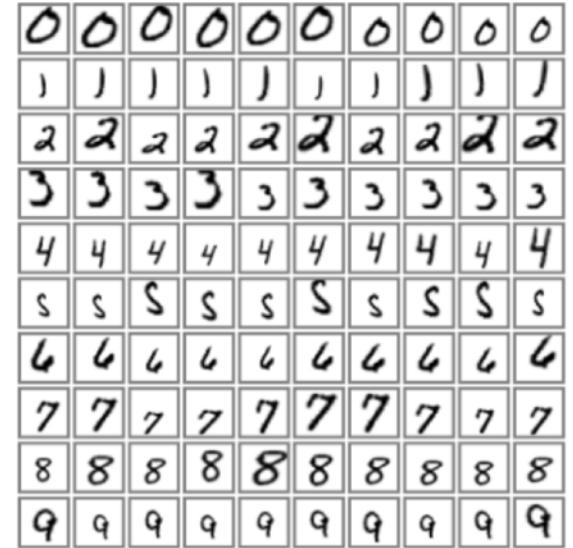
# What is Machine Learning?

- From Data to Knowledge



# A Standard Example of ML

- The MNIST (Modified NIST) database of hand-written digits recognition
  - Publicly available
  - A huge amount about how well various ML methods do on it
  - 60,000 + 10,000 hand-written digits (28x28 pixels each)



Very hard to say what makes a 2

0 0 0 1 1 1 1 1 1 2

2 2 2 2 2 2 2 3 3 3

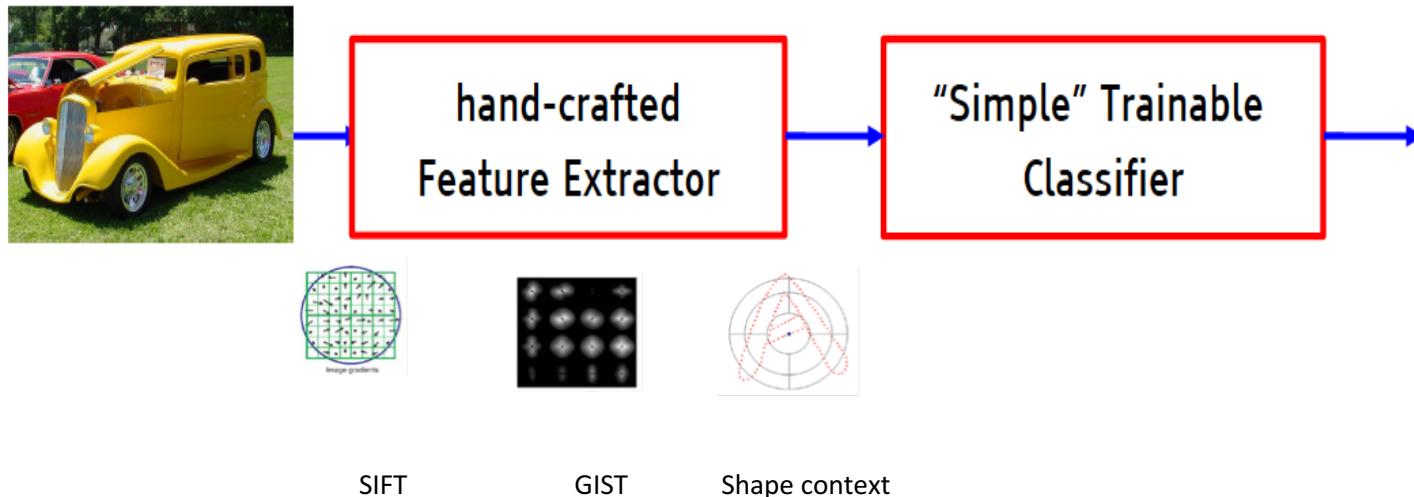
3 4 4 4 4 4 5 5 5 5

6 6 7 7 7 7 7 8 8 8

8 8 9 9 9 9 9 9 9

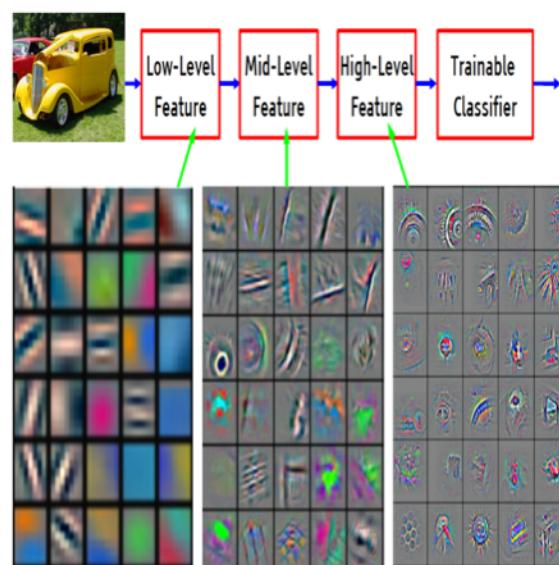
# Traditional Model (before 2012)

- Fixed/engineered features + trainable classifier (分类器)
  - Designing a feature extractor requires considerable efforts by experts



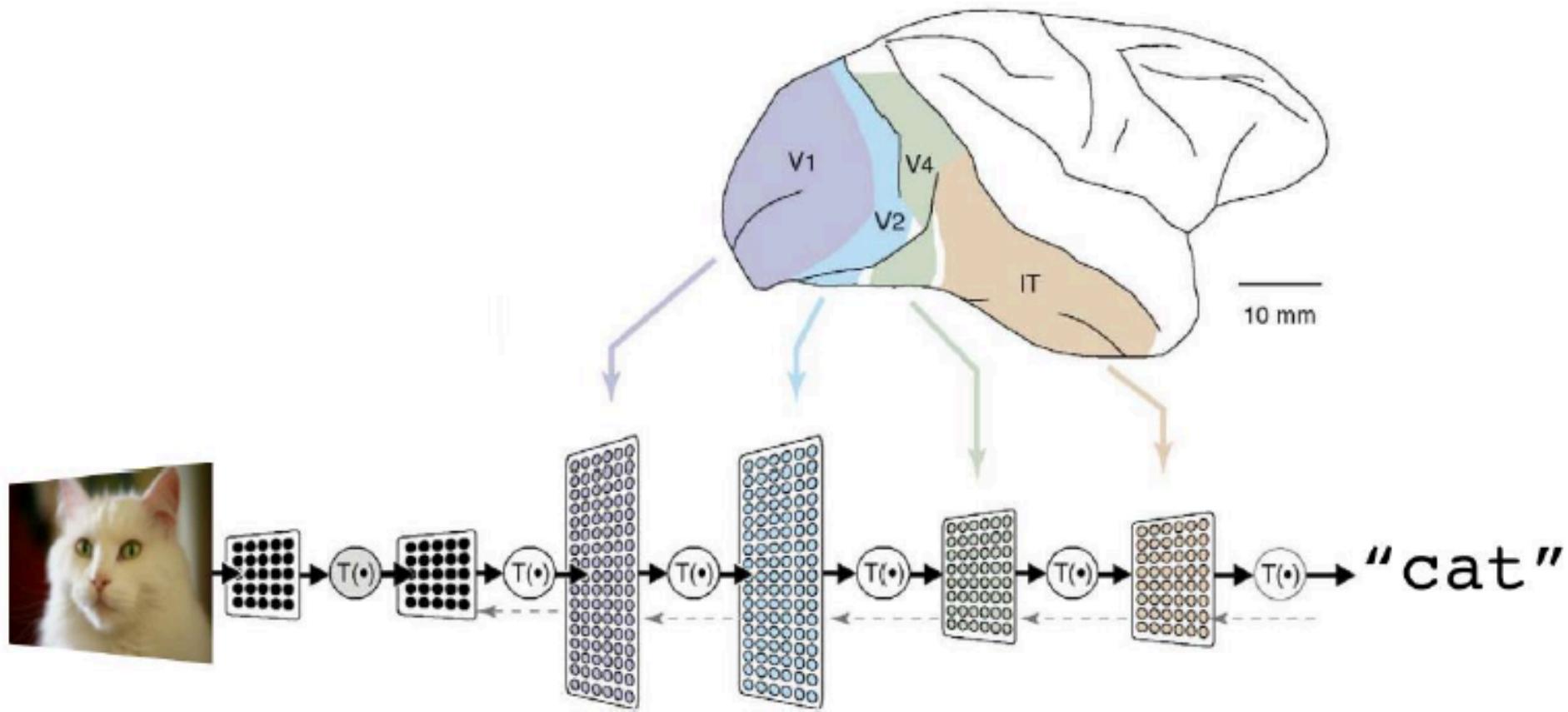
# Deep Learning (after 2012)

- Learning Hierarchical Representations
- DEEP means **more than one** stage of **non-linear** feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Deep Learning Architecture



# Deep Learning is Not New

- 1980s technology (Neural Networks)

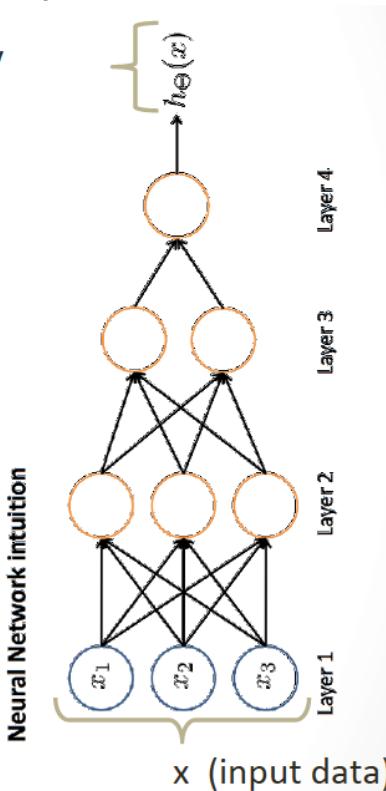
Supervised learning

- Given  $x$  and  $y$ , learn  $p(y|x)$
- Is this photo,  $x$ , a “cat”,  $y$ ?



$x =$

(label)  $y$



# About Neural Networks

- Pros
  - Simple to learn  $p(y|x)$
  - Results OK for shallow nets
- Cons
  - Does not learn  $p(x)$
  - Trouble with  $> 3$  layers
  - Overfits
  - Slow to train

# Deep Learning beats NN

- Pros
  - Simple to learn  $p(y|x)$
  - Results OK for shallow nets
- Cons
  - Does not learn  $p(x)$
  - Trouble with  $> 3$  layers
  - Overfitting
  - Slow to train

Unsupervised feature learning: RMBs, DAEs,  
...

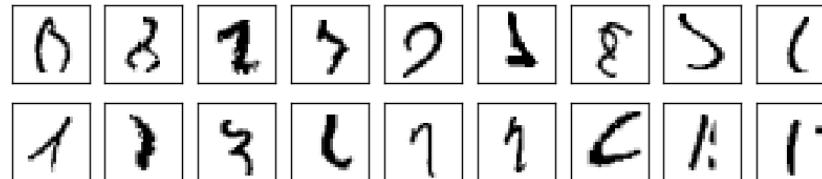
- New activation functions: ReLU, ...
- Gated mechanism

- Dropout
- Maxout
- Stochastic Pooling

GPU

# Results on MNIST

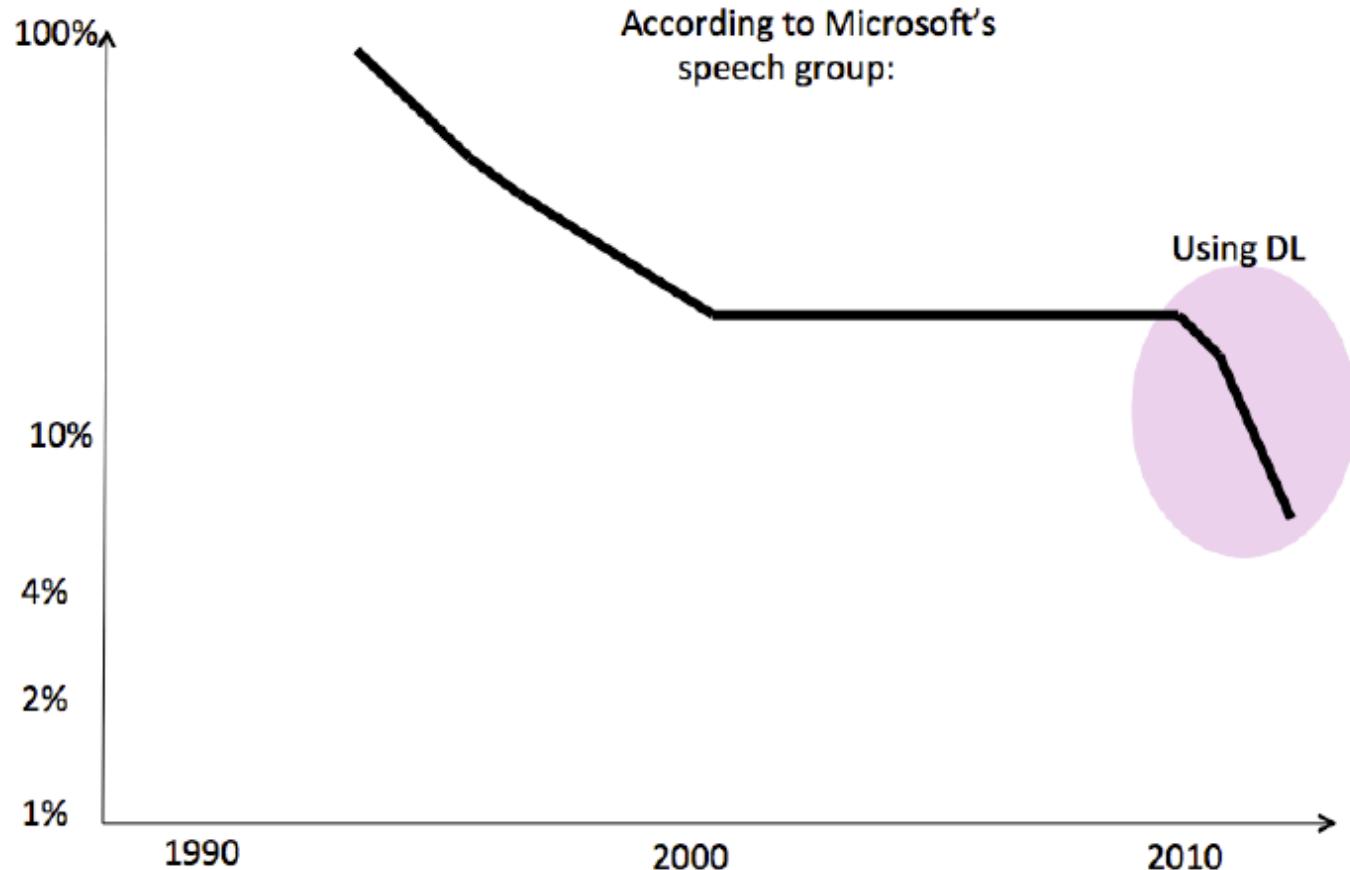
- Naïve Neural Network
  - 96.59%
- SVM (default settings for libsvm)
  - 94.35%
- Optimal SVM [Andreas Mueller]
  - 98.56%
- The state of the art: Convolutional NN (2013)
  - 99.79%



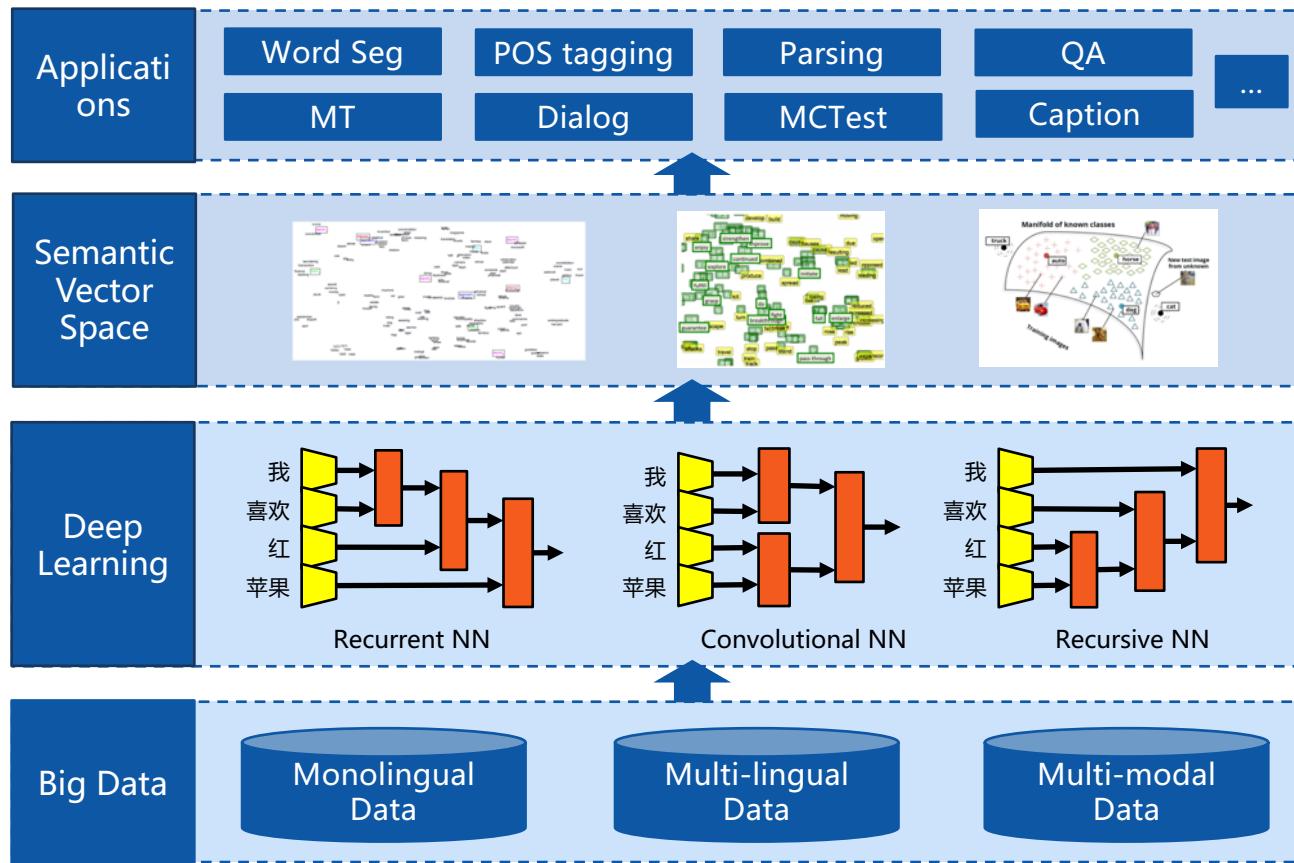
# Deep Learning Wins

9. MICCAI 2013 Grand Challenge on Mitosis Detection
8. ICPR 2012 Contest on Mitosis Detection in Breast Cancer Histological Images
7. ISBI 2012 Brain Image Segmentation Challenge (with superhuman pixel error rate)
6. IJCNN 2011 Traffic Sign Recognition Competition (only our method achieved superhuman results)
5. ICDAR 2011 offline Chinese Handwriting Competition
4. Online German Traffic Sign Recognition Contest
3. ICDAR 2009 Arabic Connected Handwriting Competition
2. ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition
1. ICDAR 2009 French Connected Handwriting Competition. Compare the overview page on handwriting recognition.
  - <http://people.idsia.ch/~juergen/deeplearning.html>

# Deep Learning for Speech Recognition



# Deep Learning for NLP

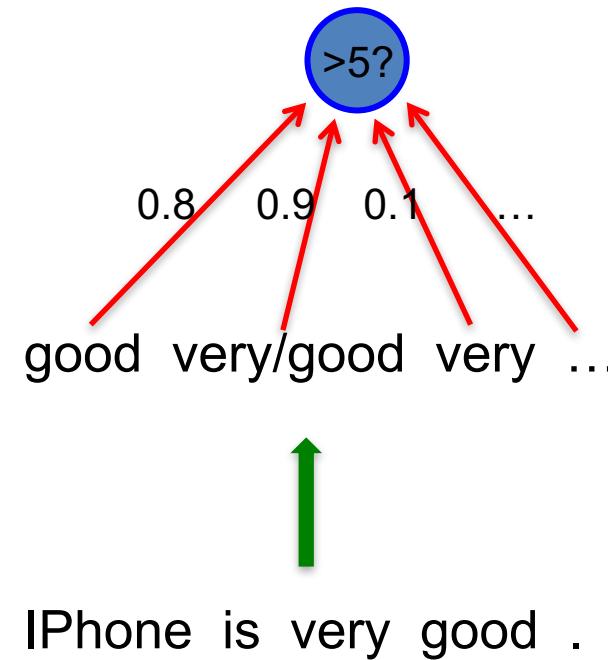
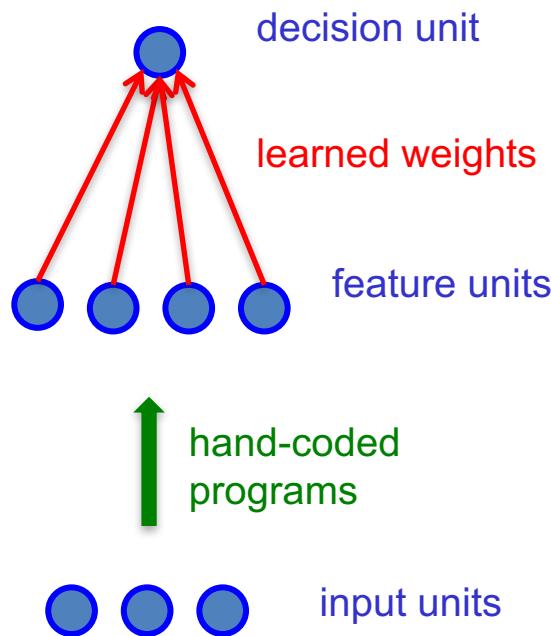


## Part 2.2: Feedforward Neural Networks

# The Traditional Paradigm for ML

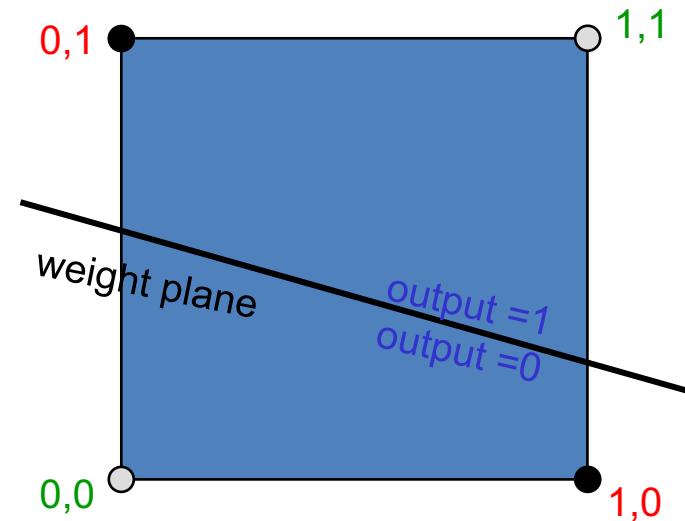
1. Convert the raw input vector into a vector of feature activations
  - Use hand-written programs based on common-sense to define the features
2. Learn how to weight each of the feature activations to get a single scalar quantity
3. If this quantity is above some threshold, decide that the input vector is a positive example of the target class

# The Standard Perceptron Architecture



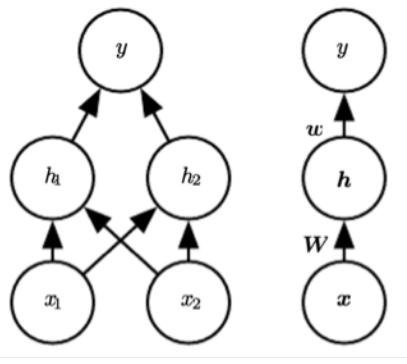
# The Limitations of Perceptrons

- The **hand-coded features**
  - Great influence on the performance
  - Need lots of cost to find suitable features
- A **linear classifier** with a hyperplane
  - Cannot separate non-linear data, such as XOR function cannot be learned by a single-layer perceptron



The **positive** and **negative** cases cannot be separated by a plane

# Learning with Non-linear Hidden Layers



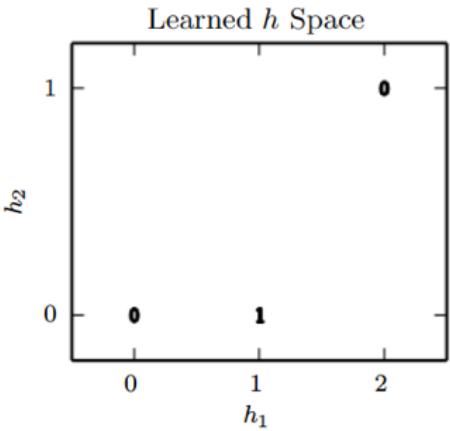
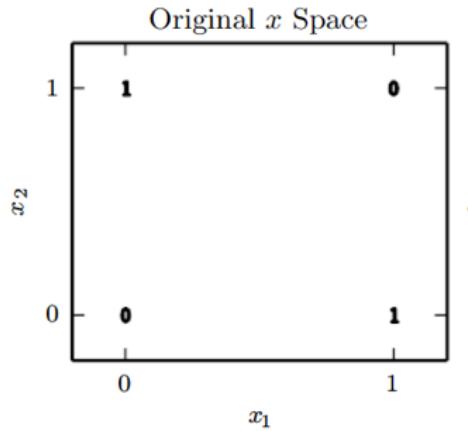
$$\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$\mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

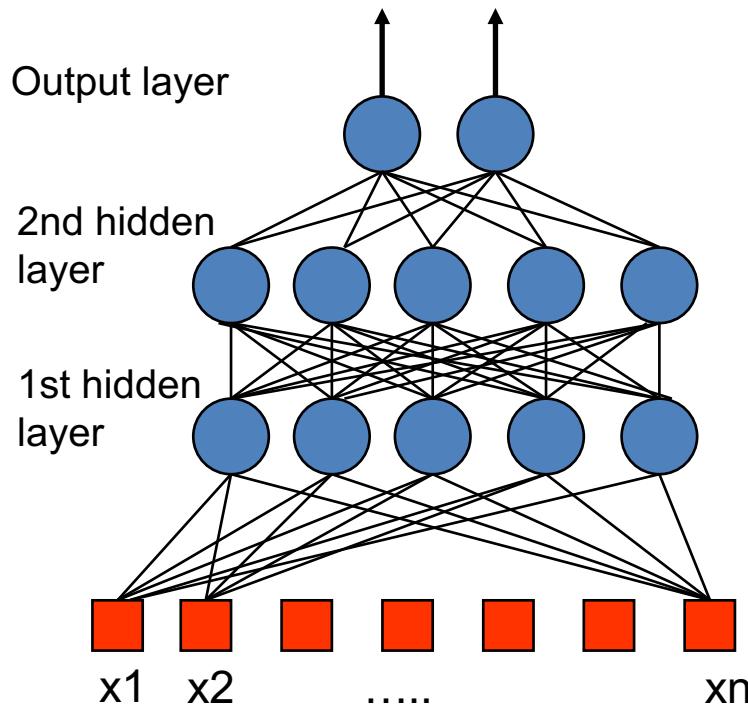
$$\mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix},$$

$$b = 0.$$

$$f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = \mathbf{w}^\top \max\{0, \mathbf{W}^\top \mathbf{x} + \mathbf{c}\} + b.$$

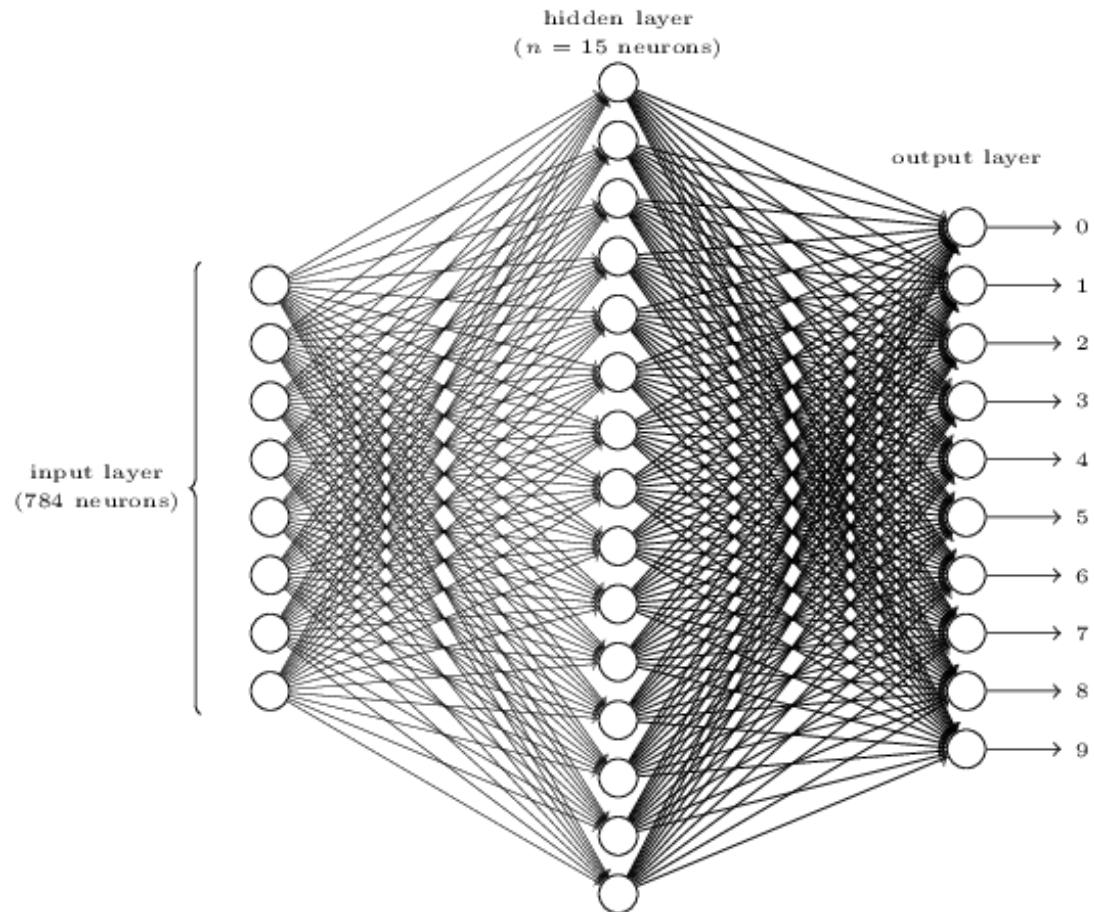


# Feedforward Neural Networks

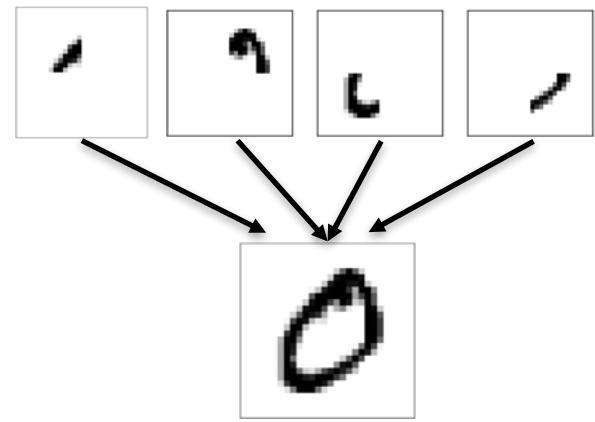


- The information is propagated from the inputs to the outputs
- Time has no role (NO cycle between outputs and inputs)
- Multi-layer Perceptron (**MLP**)?
- Learning the weights of hidden units is equivalent to **learning features**
- Networks without hidden layers are very limited in the input-output mappings
  - More layers of linear units do not help. It's still linear
  - Fixed output non-linearities are not enough

# Multiple Layer Neural Networks



- What are those hidden neurons doing?
  - Maybe represent outlines



# General Optimizing (Learning) Algorithms

- Gradient Descent

$$\theta \leftarrow \theta + \epsilon \nabla_{\theta} \sum_t L(f(\mathbf{x}^{(t)}; \theta), \mathbf{y}^{(t)}; \theta)$$

- Stochastic Gradient Descent (SGD)

- Minibatch SGD ( $m > 1$ ), Online GD ( $m = 1$ )

---

**Algorithm 8.1** Stochastic gradient descent (SGD) update at training iteration  $k$

---

**Require:** Learning rate  $\epsilon_k$ .

**Require:** Initial parameter  $\theta$

**while** stopping criterion not met **do**

        Sample a minibatch of  $m$  examples from the training set  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  with corresponding targets  $\mathbf{y}^{(i)}$ .

        Compute gradient estimate:  $\hat{\mathbf{g}} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

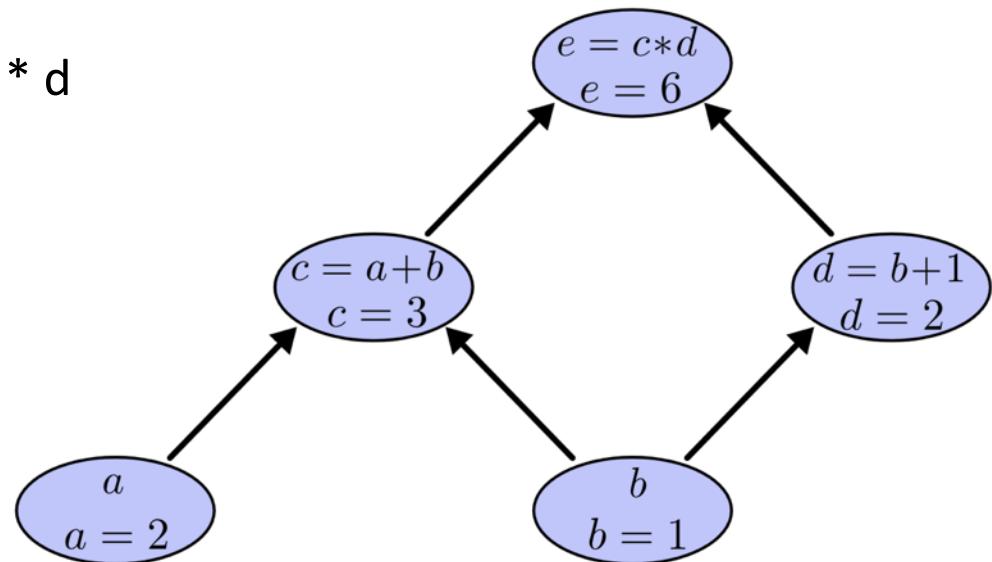
        Apply update:  $\theta \leftarrow \theta - \epsilon \hat{\mathbf{g}}$

**end while**

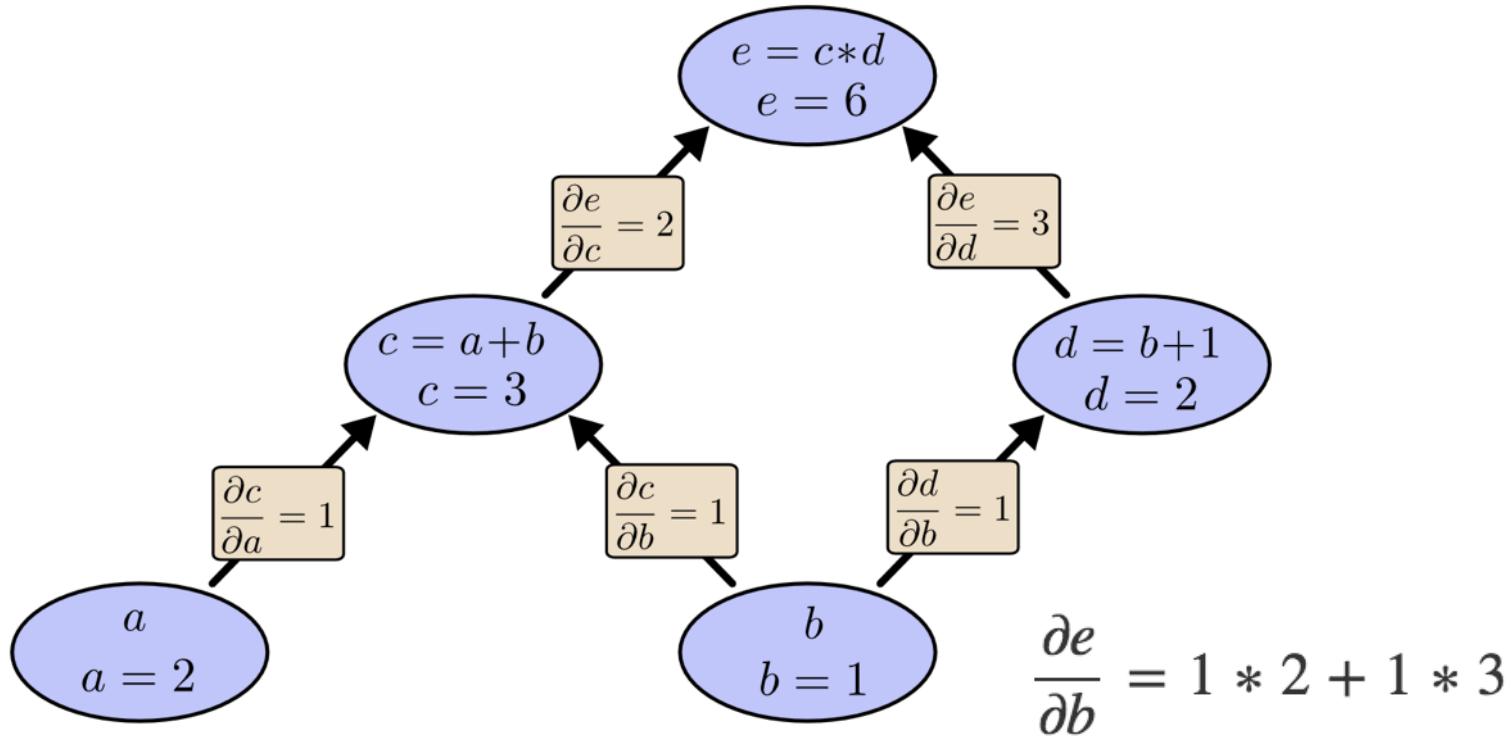
---

# Computational/Flow Graphs

- Describing Mathematical Expressions
- For example
  - $e = (a + b) * (b + 1)$ 
    - $c = a + b$ ,  $d = b + 1$ ,  $e = c * d$
  - If  $a = 2$ ,  $b = 1$

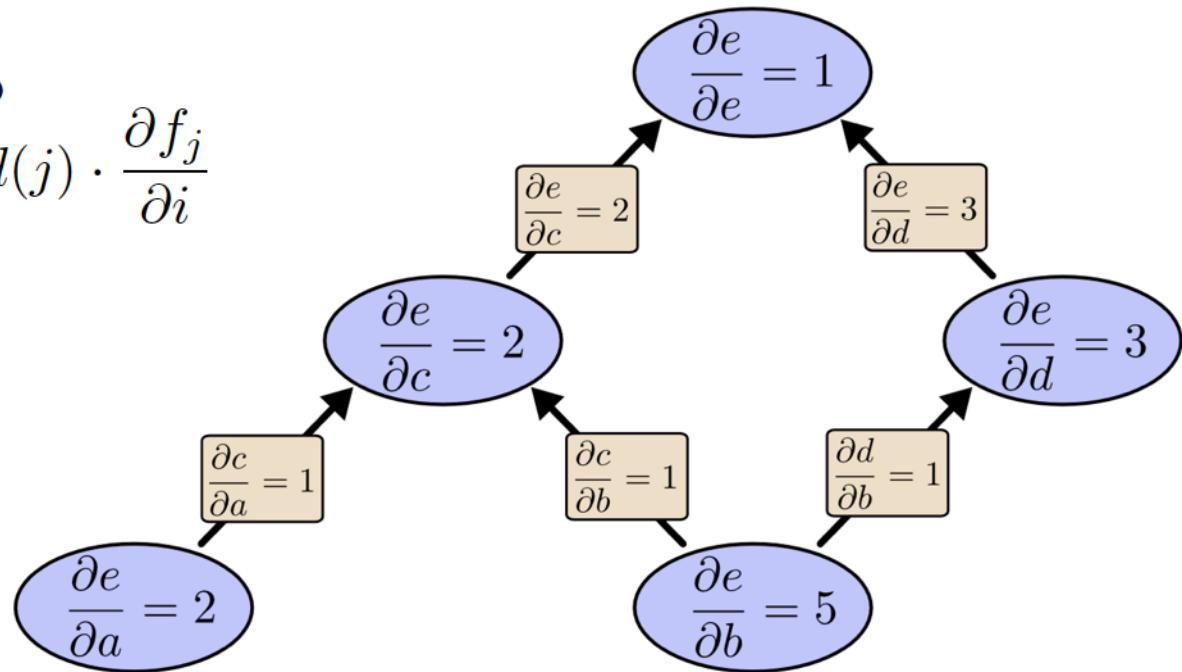


# Derivatives on Computational Graphs



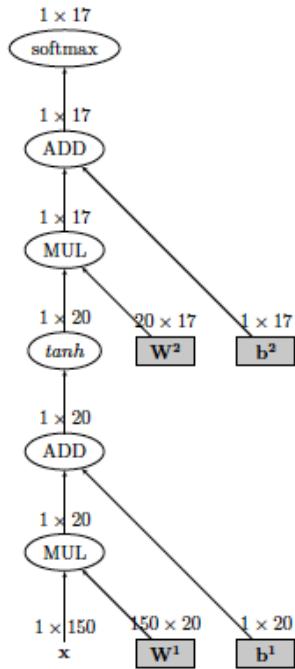
# Computational Graph Backward Pass (Backpropagation)

```
1:  $d(N) \leftarrow 1$ 
2: for  $i = N-1$  to 1 do
3:    $d(i) \leftarrow \sum_{j \in \pi(i)} d(j) \cdot \frac{\partial f_j}{\partial i}$ 
```

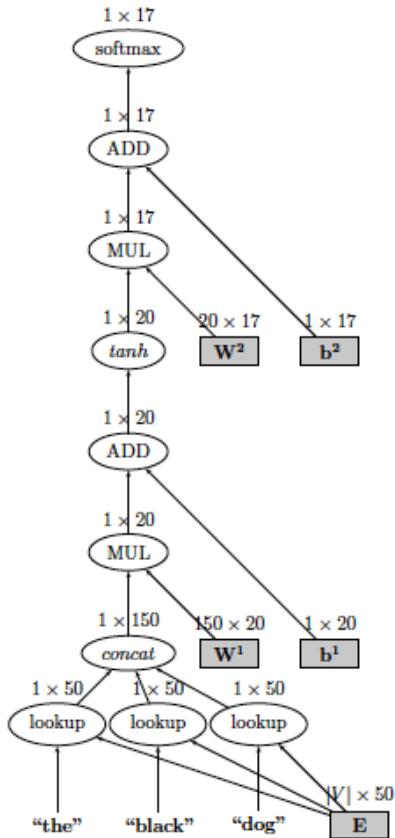


# An FNN POS Tagger

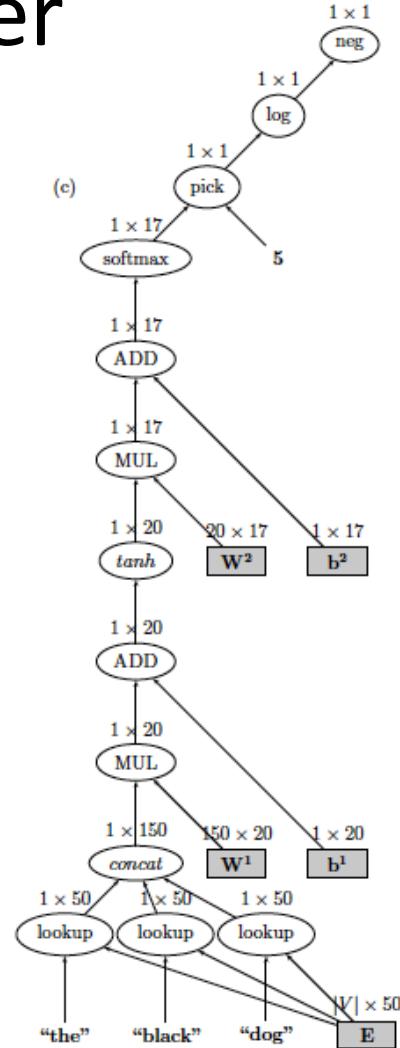
(a)



(b)



(c)



## Part 2.3: Word Embeddings

# Typical Approaches for Word Representation

- 1-hot representation (orthogonality)
  - bag-of-word model

**star** [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ...]

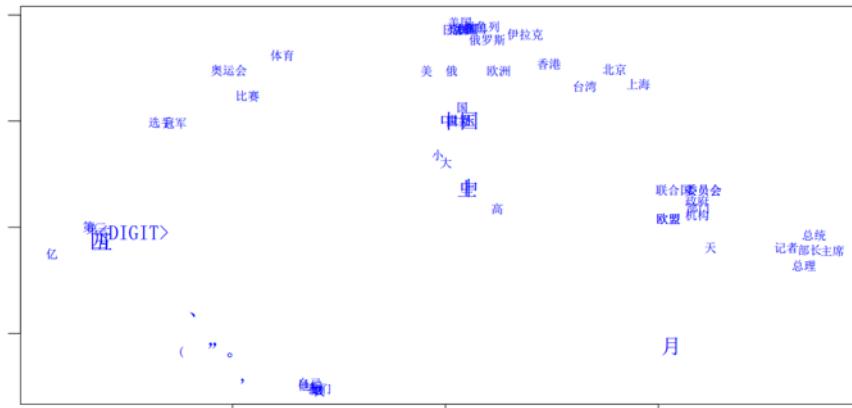
**sun** [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ...]

$\text{sim(star, sun)} = 0$

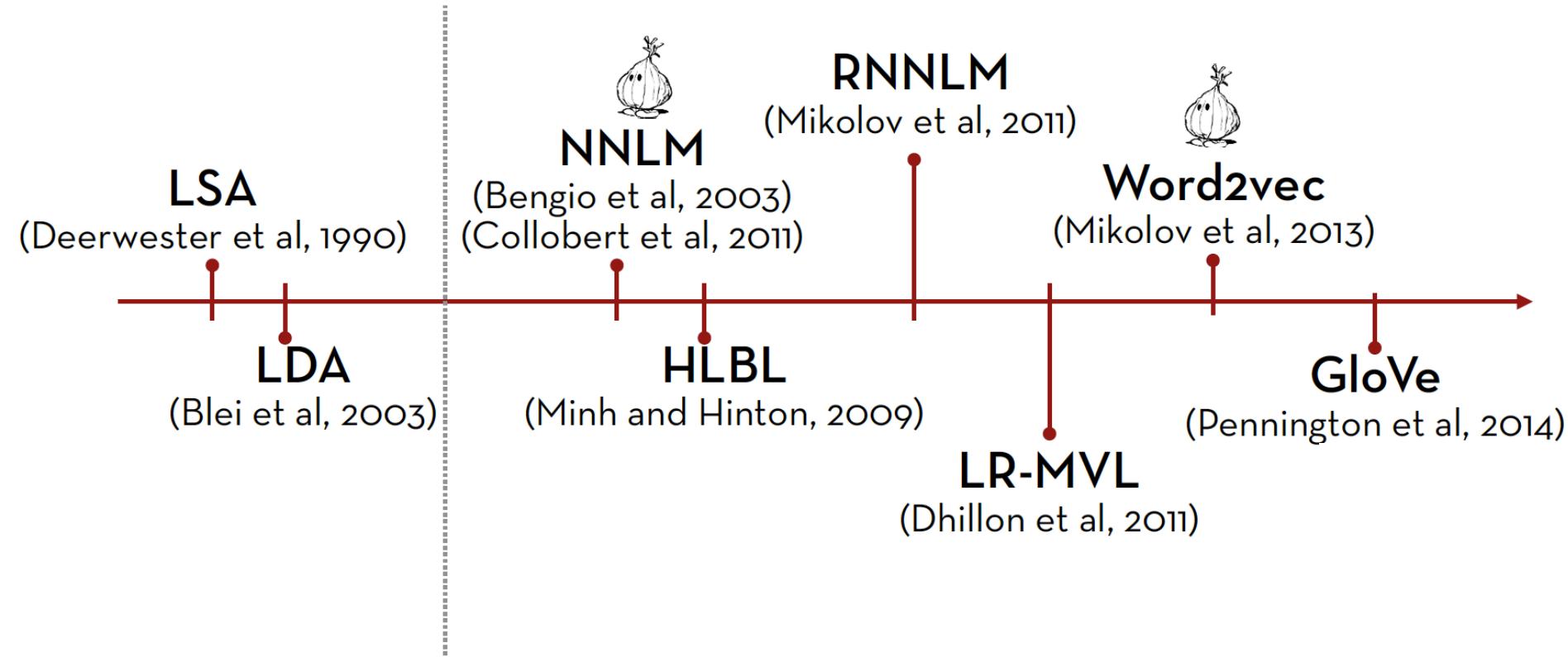


# Distributed Word Representation

- Each word is associated with a **low-dimension** (compressed, 50-1000), **density** (non-sparse) and **real** (continuous) vector (**word embedding**)
    - Learning word vectors through **supervised** models
  - Nature
    - Semantic similarity as vector similarity

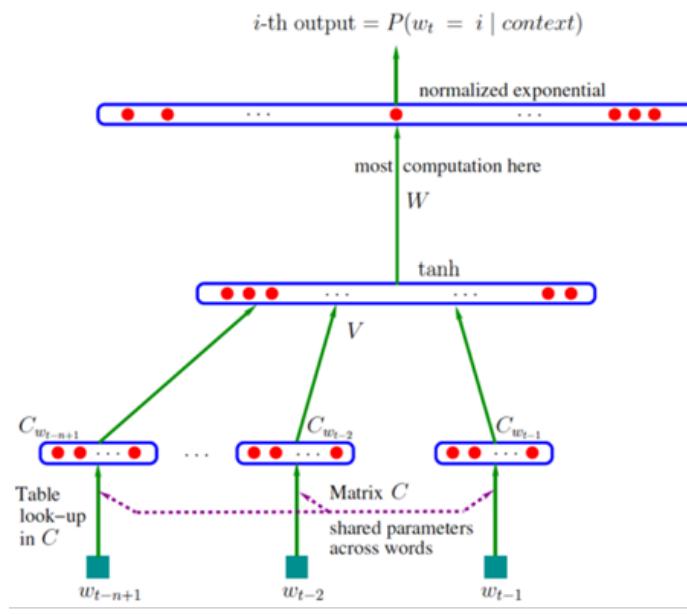


# How to obtain Word Embedding



# Neural Network Language Models

- Neural Network Language Models (NNLM)
  - Feed Forward (Bengio et al. 2003)



- **Maximum-Likelihood Estimation**
- Back-propagation
- Input:  $(n - 1)$  embeddings

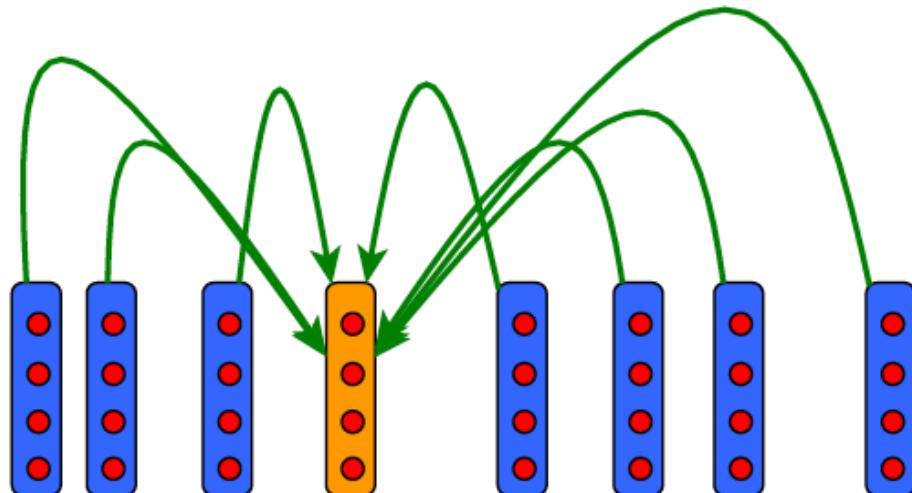
$$P(w_t = k | w_{t-n+1}, \dots, w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^N e^{a_l}}$$

$$a_k = b_k + \sum_{i=1}^h W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j)$$

$$L(\theta) = \sum_t \log P(w_t | w_{t-n+1}, \dots, w_{t-1})$$

# Predict Word Vector Directly

- SENNA (Collobert and Weston, 2008)
- word2vec (Mikolov et al. 2013)

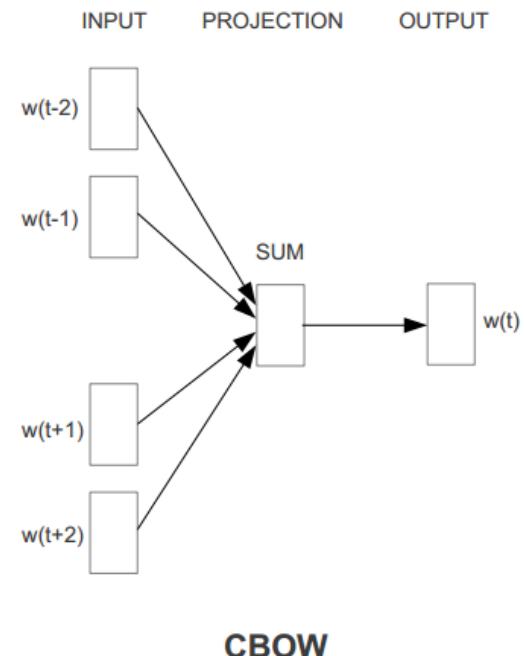


I got the shotgun. You got the briefcase.

# Word2vec: CBOW (Continuous Bag-of-Word)

- Add inputs from words within short window to predict the current word
- The weights for different positions are shared
- Computationally much more efficient than normal NNLM
- The hidden layer is just linear
- Each word is an embedding  $v(w)$
- Each context is an embedding  $v'(c)$

$$r(c) = v'(c_{-2}) + v'(c_{-1}) + v'(c_1) + v'(c_2)$$



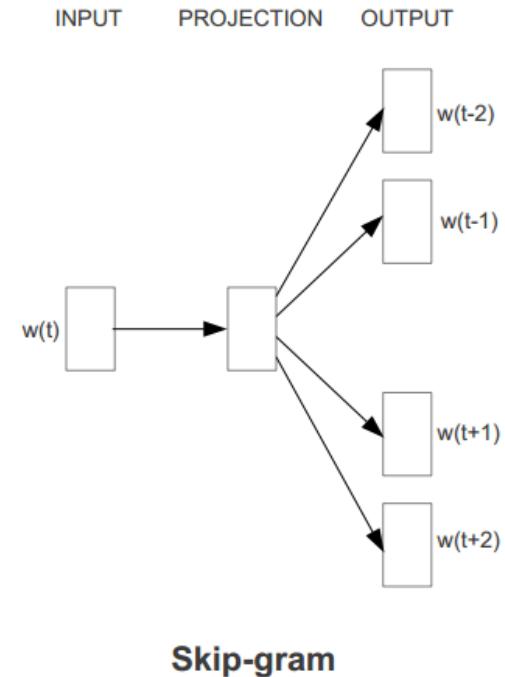
$$p(v(w) | r(c)) = \frac{\exp(r(c) \cdot v(w))}{\sum_{w^*} \exp(r(c) \cdot v(w^*))}$$

# Word2vec: Skip-Gram

- Predicting surrounding words using the current word
- Similar performance with CBOW
- Each word is an embedding  $v(w)$
- Each context is an embedding  $v'(c)$

$$\frac{1}{|\mathcal{C}|} \sum_{(w,c) \in \mathcal{C}} \log p(v'(c) | v(w))$$

$$p(v'(c) | v(w)) = \frac{\exp(v'(c) \cdot v(w))}{\sum_{c^*} \exp(v'(c^*) \cdot v(w))}$$

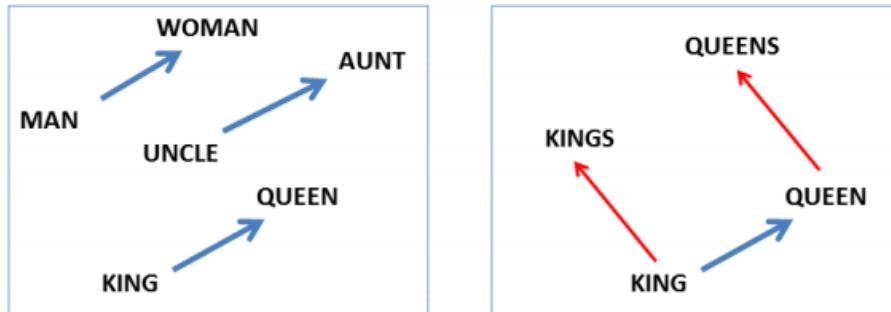


# Word2vec Training

- SGD + backpropagation
- Most of the computational cost is a function of the size of the vocabulary (millions)
- Training accelerating
  - Negative Sampling
    - Mikolov et al. 2013
  - Hierarchical Decomposition
    - Morin and Bengio 2005. Mnih and Hinton 2008. Mikolov et al. 2013
  - Graph Processing Unit (GPU)

# Word Analogy

$$v(\text{king}) - v(\text{queen}) \approx v(\text{man}) - v(\text{woman})$$



## Part 2.4: Recurrent and Other Neural Networks

# Language Models

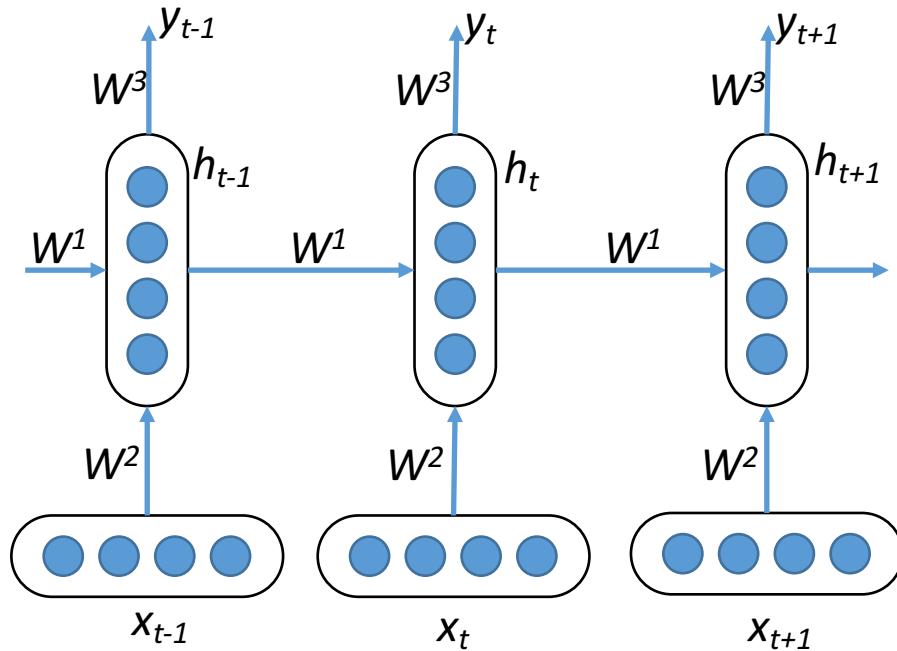
- A language model computes a probability for a sequence of words:  $P(w_1, \dots, w_n)$  or predicts a probability for the next word:  $P(w_{n+1} | w_1, \dots, w_n)$
- Useful for machine translation, speech recognition, and so on
  - Word ordering
    - $P(\text{the cat is small}) > P(\text{small the is cat})$
  - Word choice
    - $P(\text{there are four cats}) > P(\text{there are for cats})$

# Traditional Language Models

- An incorrect but necessary Markov assumption!
  - Probability is usually conditioned on window of  $n$  previous words
  - $P(w_1, \dots, w_n) = \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1}) \approx \prod_{i=1}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1})$
- How to estimate probabilities
  - $p(w_2 | w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$     $p(w_3 | w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$
- Performance improves with keeping around higher n-grams counts and doing smoothing, such as backoff (e.g. if 4-gram not found, try 3-gram, etc)
- Disadvantages
  - There are A LOT of n-grams!
  - Cannot see too long history
    - P(坐/作 了一整天的 火车/作业)

# Recurrent Neural Networks (RNNs)

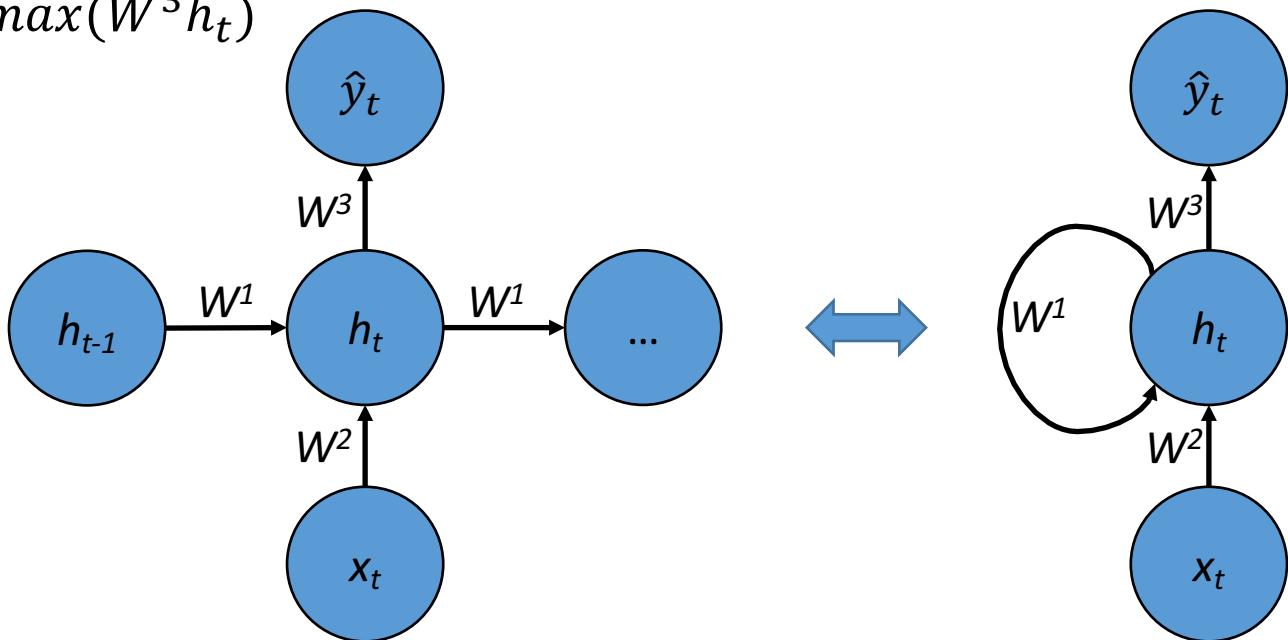
- Condition the neural network on all previous inputs
- RAM requirement only scales with number of inputs



# Recurrent Neural Networks (RNNs)

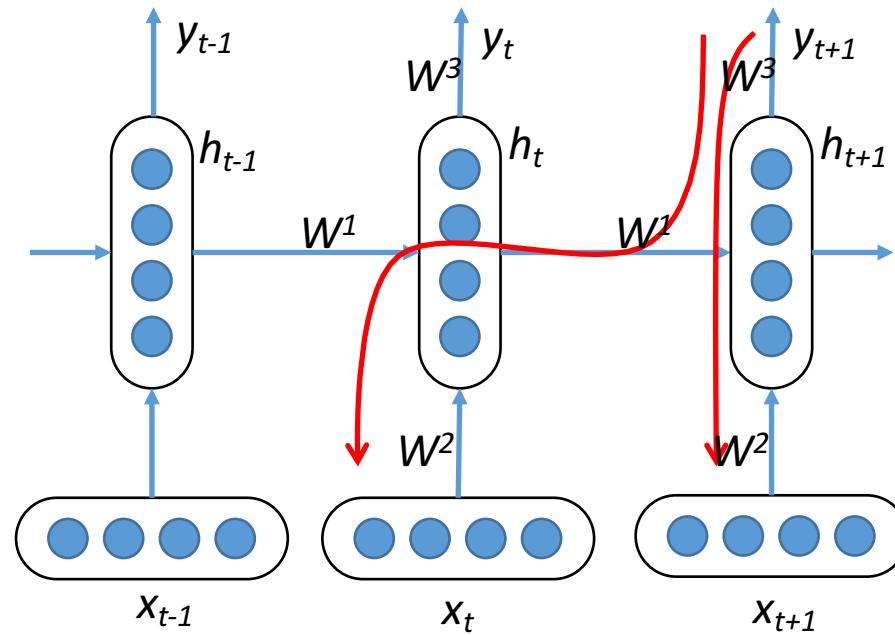
- At a single time step  $t$

- $h_t = \tanh(W^1 h_{t-1} + W^2 x_t)$
- $\hat{y}_t = \text{softmax}(W^3 h_t)$



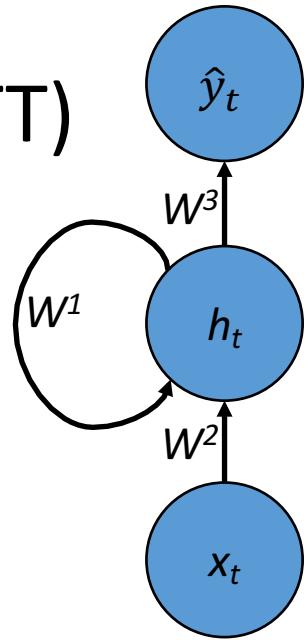
# Training RNNs is hard

- Ideally inputs from many time steps ago can modify output  $y$
- For example, with 2 time steps



# BackPropagation Through Time (BPTT)

- Total error is the sum of each error at time step  $t$ 
  - $\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W}$
  - $\frac{\partial E_t}{\partial W^3} = \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial W^3}$  is easy to be calculated
  - But to calculate  $\frac{\partial E_t}{\partial W^1} = \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial W^1}$  is hard (also for  $W^2$ )
  - Because  $h_t = \tanh(W^1 h_{t-1} + W^2 x_t)$  depends on  $h_{t-1}$ , which depends on  $W^1$  and  $h_{t-2}$ , and so on.
  - So  $\frac{\partial E_t}{\partial W^1} = \sum_{k=1}^t \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W^1}$



# The vanishing gradient problem

- $\frac{\partial E_t}{\partial W} = \sum_{k=1}^t \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \boxed{\frac{\partial h_k}{\partial W}}, h_t = \tanh(W^1 h_{t-1} + W^2 x_t)$
- $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^t W^1 \text{diag}[\tanh'(\dots)]$
- $\left\| \frac{\partial h_t}{\partial h_{t-1}} \right\| \leq \gamma \|W^1\| \leq \gamma \lambda_1$ 
  - where  $\gamma$  is bound  $\|\text{diag}[\tanh'(\dots)]\|$ ,  $\lambda_1$  is the largest singular value of  $W^1$
- $\left\| \frac{\partial h_t}{\partial h_k} \right\| \leq (\gamma \lambda_1)^{t-k} \rightarrow 0$ , if  $\gamma \lambda_1 < 1$
- This can become very small or very large quickly  $\rightarrow$  Vanishing or exploding gradient
  - Trick for exploding gradient: clipping trick (set a threshold)

# A “solution”

- Intuition
  - Ensure  $\gamma\lambda_1 \geq 1 \rightarrow$  to prevent vanishing gradients
- So ...
  - Proper initialization of the W
  - To use ReLU instead of tanh or sigmoid activation functions

# A better “solution”

- Recall the original transition equation

- $h_t = \tanh(W^1 h_{t-1} + W^2 x_t)$

- We can instead update the state **additively**

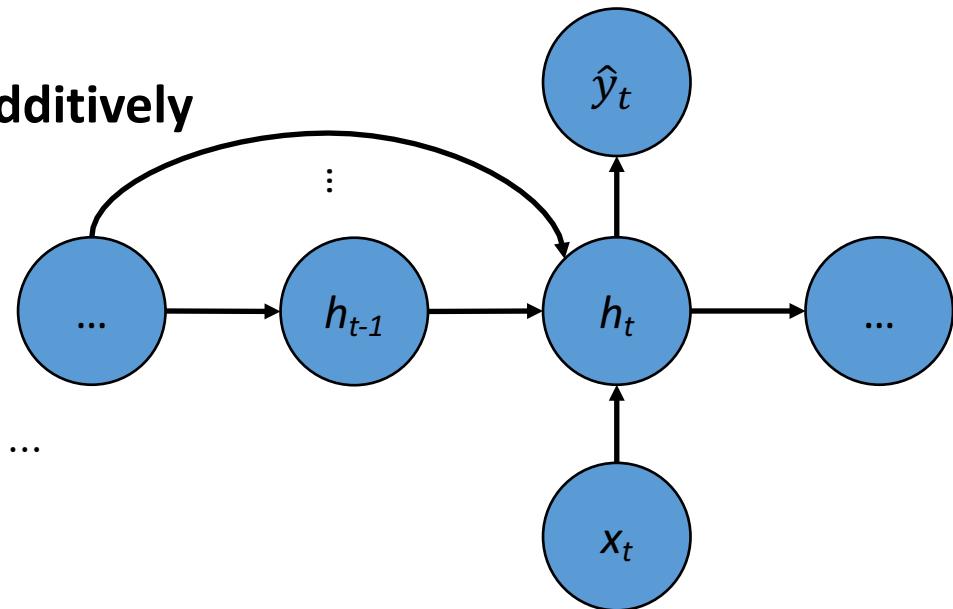
- $u_t = \tanh(W^1 h_{t-1} + W^2 x_t)$

- $h_t = h_{t-1} + u_t$

- then,  $\left\| \frac{\partial h_t}{\partial h_{t-1}} \right\| = 1 + \left\| \frac{\partial u_t}{\partial h_{t-1}} \right\| \geq 1$

- On the other hand

- $h_t = h_{t-1} + u_t = h_{t-2} + u_{t-1} + u_t = \dots$



# A better “solution” (cont.)

- Interpolate between old state and new state (“choosing to **forget**”)
  - $f_t = \sigma(W^f x_t + U^f h_{t-1})$
  - $h_t = f_t \odot h_{t-1} + (1 - f_t) \odot u_t$
- Introduce a separate **input gate**  $i_t$ 
  - $i_t = \sigma(W^i x_t + U^i h_{t-1})$
  - $h_t = f_t \odot h_{t-1} + i_t \odot u_t$
- Selectively expose memory cell  $c_t$  with an **output gate**  $o_t$ 
  - $o_t = \sigma(W^o x_t + U^o h_{t-1})$
  - $c_t = f_t \odot c_{t-1} + i_t \odot u_t$
  - $h_t = o_t \odot \tanh(c_t)$

# Long Short-Term Memory (LSTM)

$$u_t = \tanh(W_h h_{t-1} + V_x x_t)$$

$$f_t = \text{sigmoid}(W_f h_{t-1} + V_f x_t)$$

$$i_t = \text{sigmoid}(W_i h_{t-1} + V_i x_t)$$

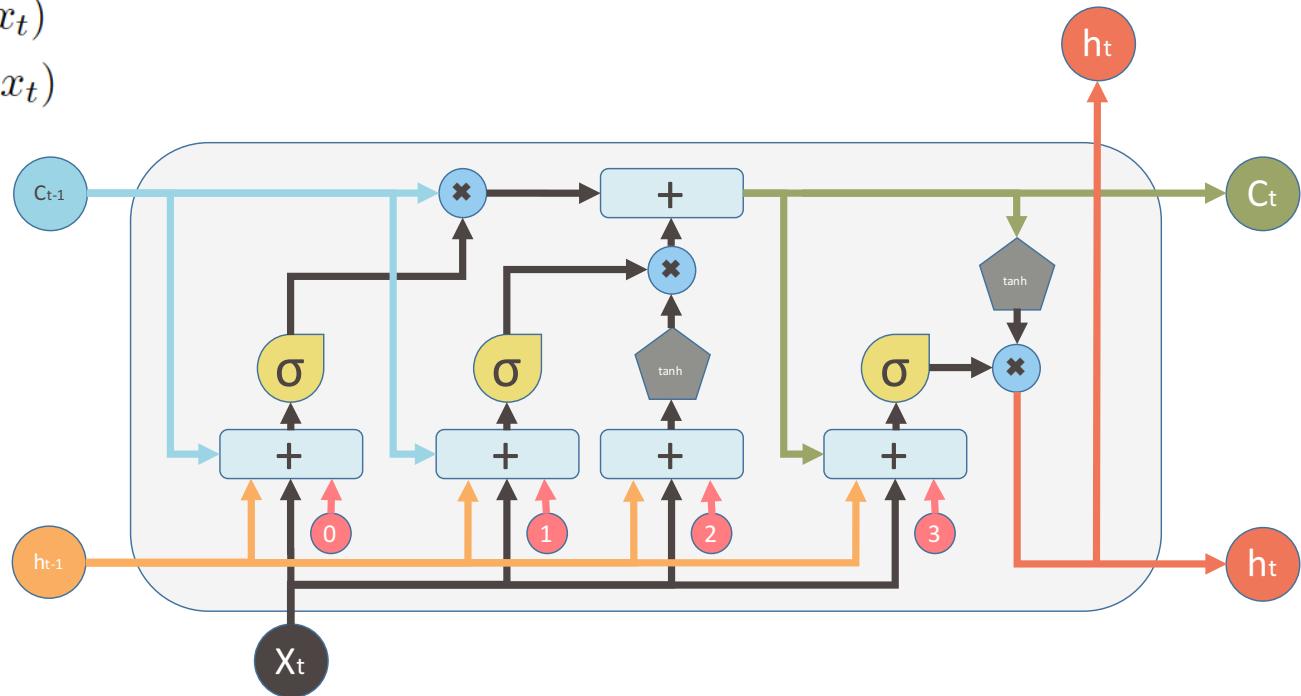
$$o_t = \text{sigmoid}(W_o h_{t-1} + V_o x_t)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

$$h_t = o_t \odot \tanh(c_t)$$

$$y_t = U h_t$$

- Hochreiter & Schmidhuber, 1997
- LSTM = additive updates + gating

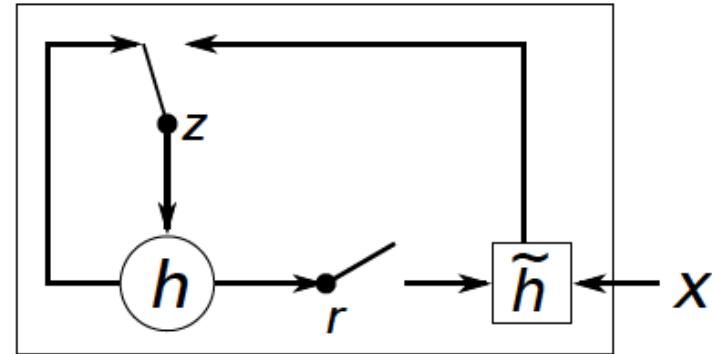


# Gated Recurrent Units, GRU (Cho et al. 2014)

- Main ideas
  - Keep around memories to capture long distance dependencies
  - Allow error messages to flow at different strengths depending on the inputs
- Update gate
  - Based on current input and hidden state
  - $z_t = \sigma(W^z x_t + U^z h_{t-1})$
- Reset gate
  - Similarly but with different weights
  - $r_t = \sigma(W^r x_t + U^r h_{t-1})$

# GRU

- New memory content
  - $\tilde{h}_t = \tanh(Wx_t + r_t \odot Uh_{t-1})$
  - Update gate  $z$  controls how much of past state should matter now
    - If  $z$  closed to 1, then we can copy information in that unit through many time steps → less vanishing gradient!
    - If reset gate  $r$  unit is close to 0, then this ignores previous memory and only stores the new input information → allows model to drop information that is irrelevant in the future
    - Units with long term dependencies have active update gates  $z$
    - Units with short-term dependencies often have rest gates  $r$  very active
  - Final memory at time step combines current and previous time steps
    - $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}$

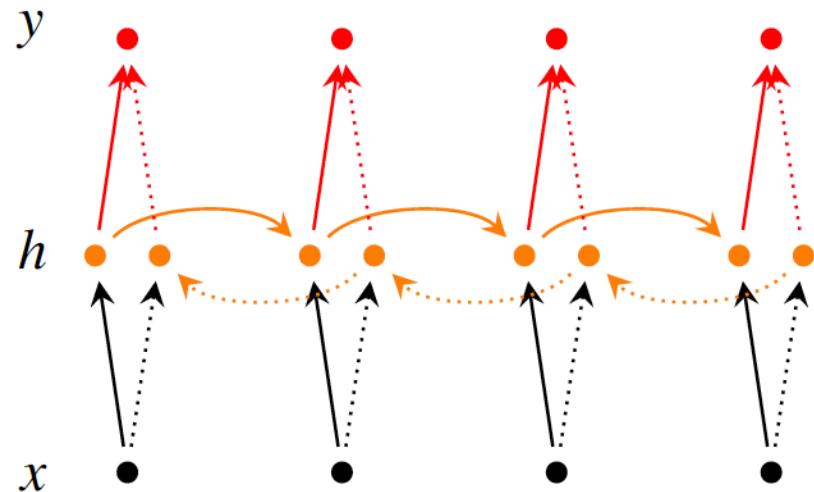


# LSTM vs. GRU

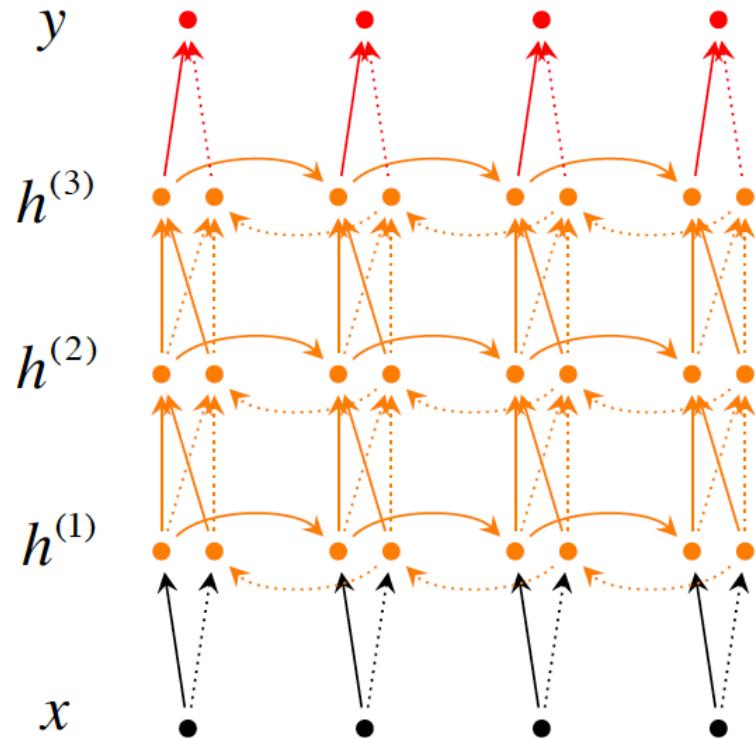
- No clear winner!
- Tuning hyperparameters like layer size is probably more important than picking the ideal architecture
- GRUs have fewer parameters and thus may train a bit faster or need less data to generalize
- If you have enough data, the greater expressive power of LSTMs may lead to better results.

# More RNNs

- Bidirectional RNN

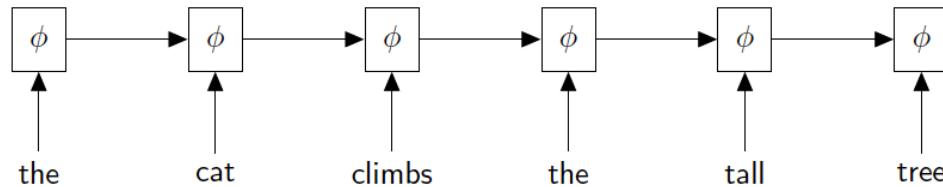


- Stack Bidirectional RNN

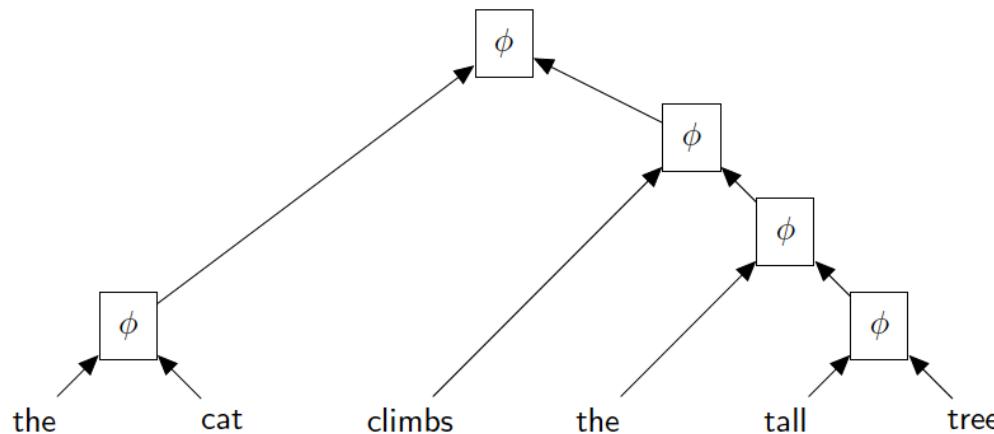


# Tree-LSTMs

- Traditional Sequential Composition



- Tree-Structured Composition

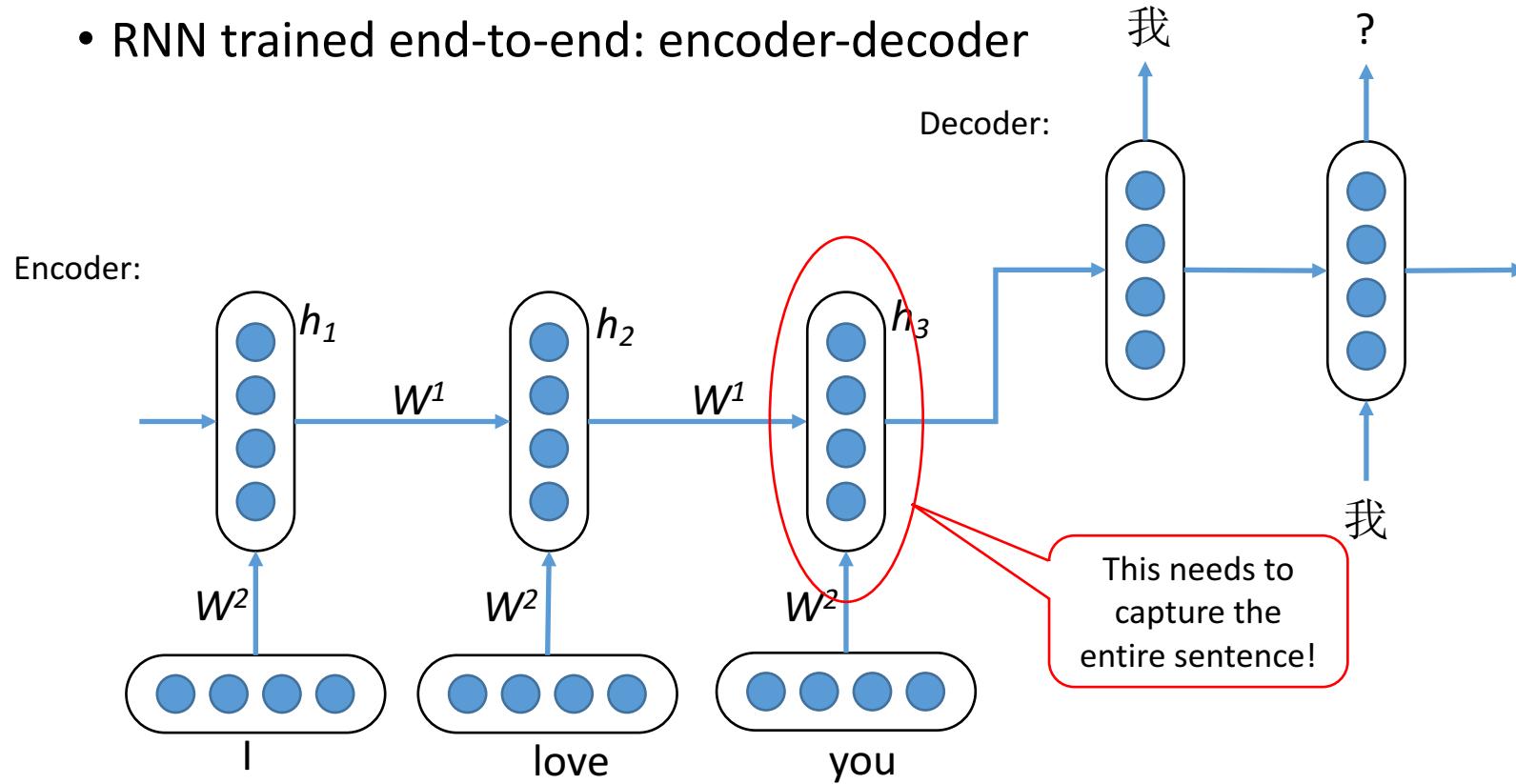


# More Applications of RNN

- Neural Machine Translation
- Handwriting Generation
- Image Caption Generation
- .....

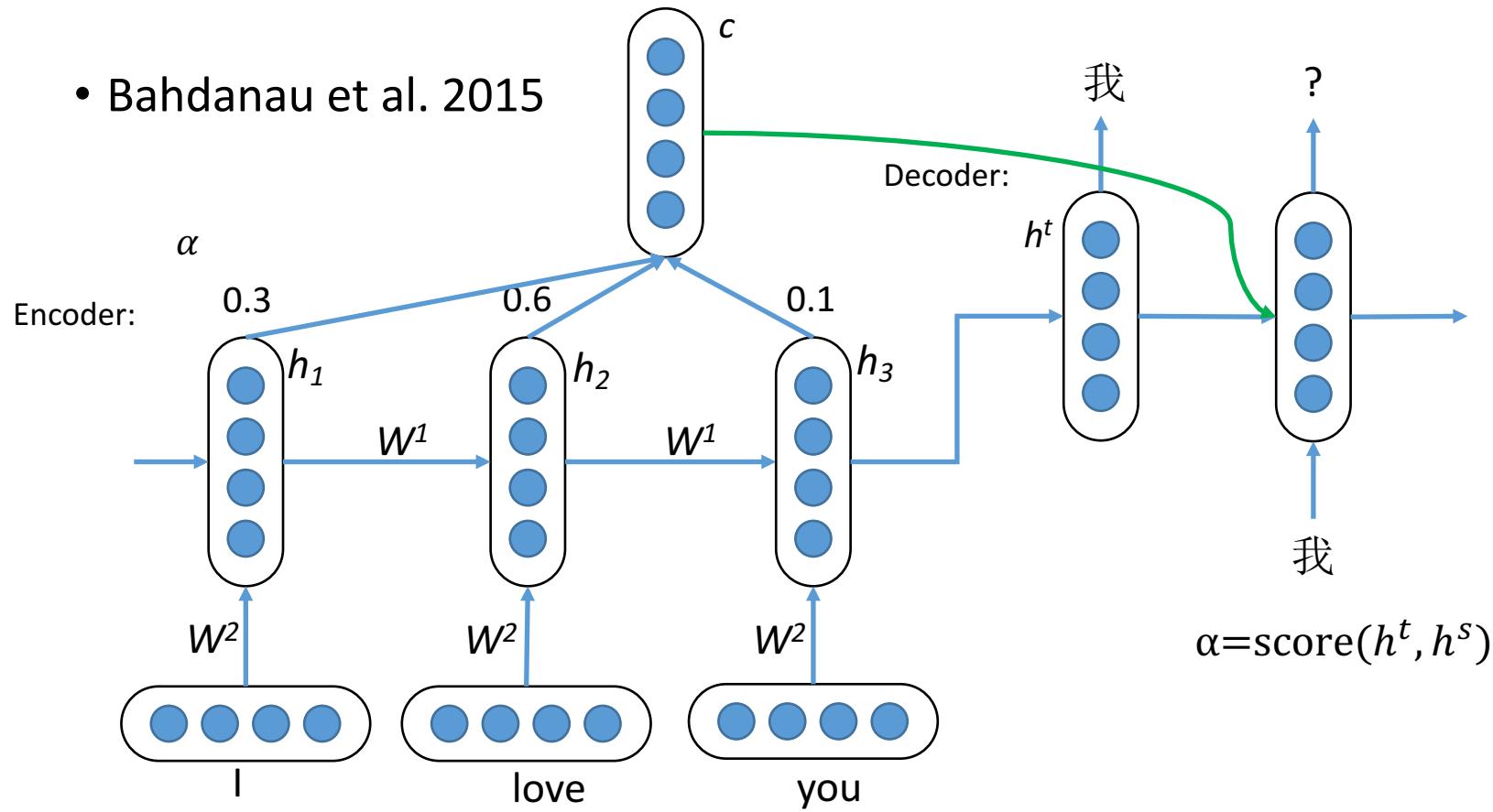
# Neural Machine Translation

- RNN trained end-to-end: encoder-decoder



# Attention Mechanism – Scoring

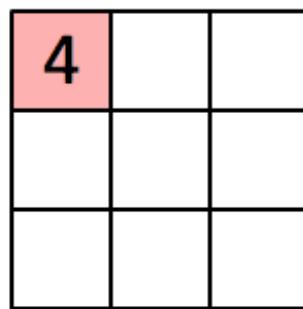
- Bahdanau et al. 2015



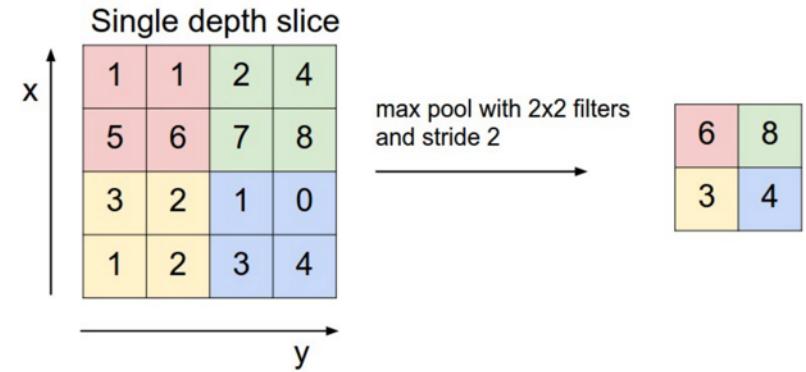
# Convolution Neural Network

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

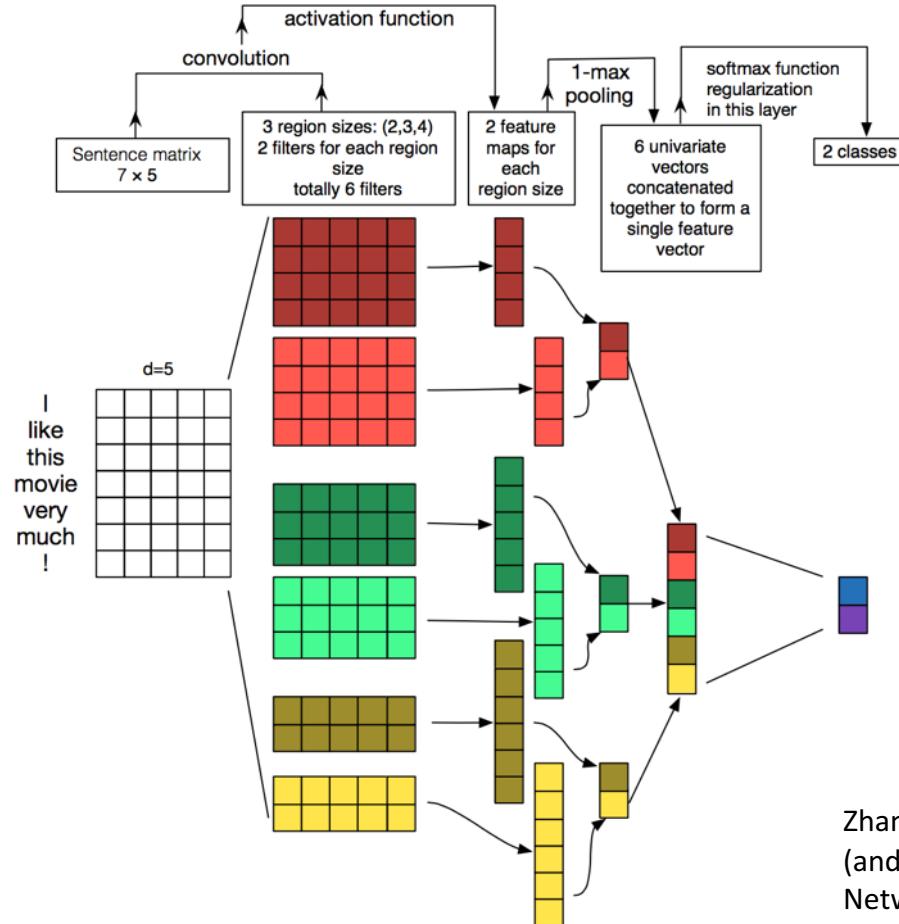


Convolved  
Feature



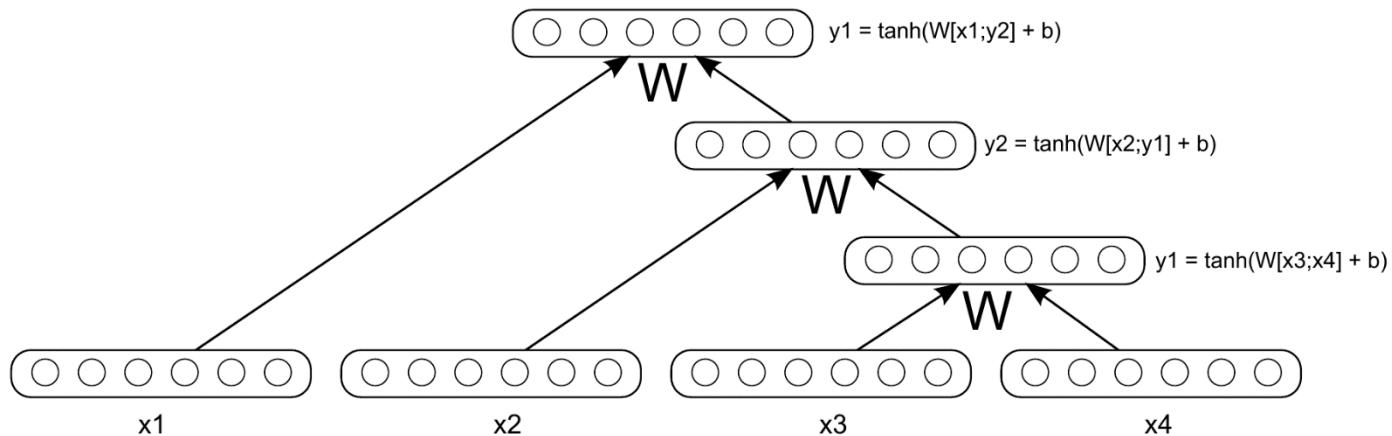
Pooling

# CNN for NLP



Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

# Recursive Neural Network



Socher, R., Manning, C., & Ng, A. (2011). Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Network. NIPS.

# Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che (HIT)

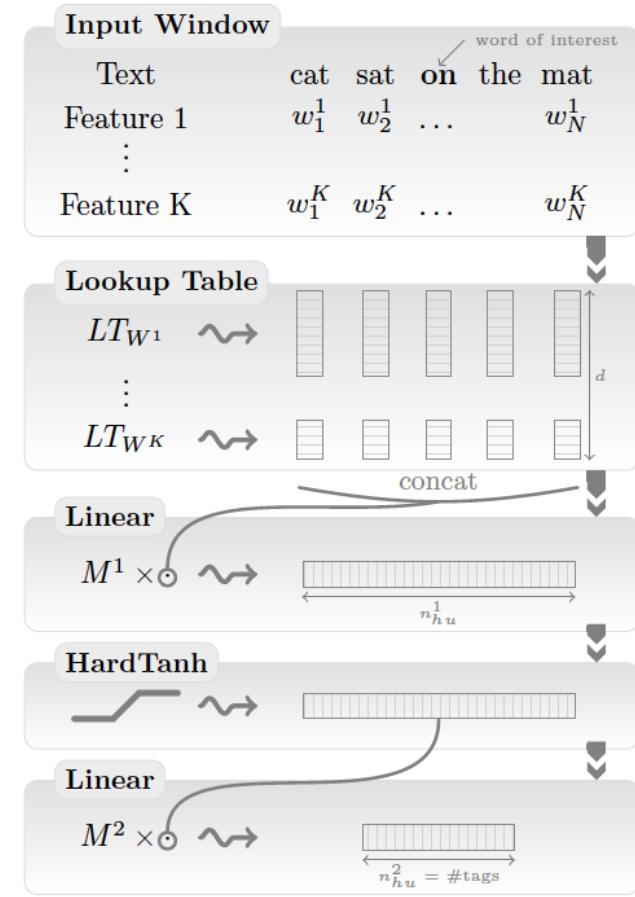
Yue Zhang (SUTD)

# Part 3: Greedy Decoding

# Part 3.1: Greedy Decoding for Tagging

# Window Approach

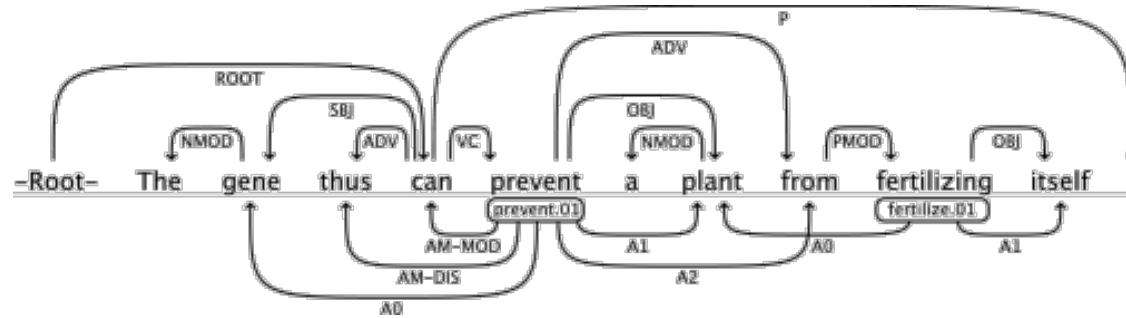
- Tasks
  - POS tagging, Chunking, NER, SRL
- Tag **one word** at a time
- Feed a **fixed-size** window of text around **each word** to tag
- Features
  - Words, POS tags, Suffix, Cascading, ...



Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* 12, 2493-2537.

# Window Approach

- Works fine for most tasks
- How to deal with **long-range dependencies**?
  - E.g. in SRL, the verb of interest might be outside the window!

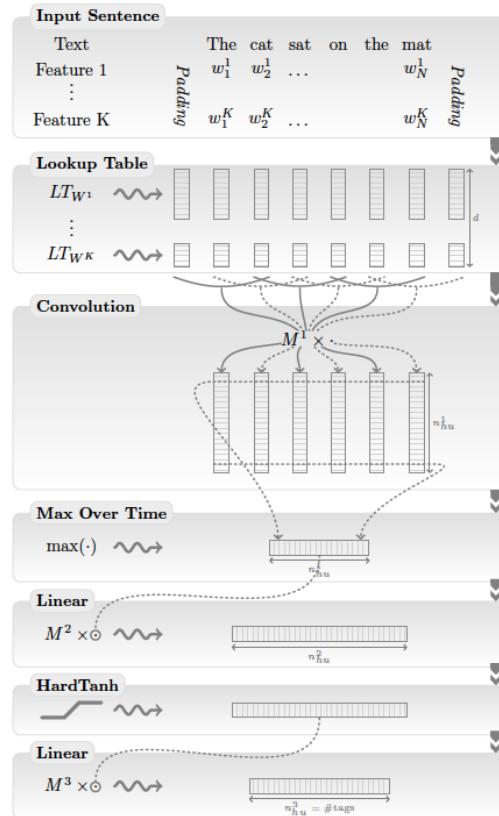


# Sentence Approach

- Tag one word at a time
  - add extra **relative position** features
- Feed the **whole sentence** to the network
- Convolutions to handle variable-length inputs
- **Max over** time to capture most relevant features
  - Outputs a fixed-sized feature vector



# Sentence Approach



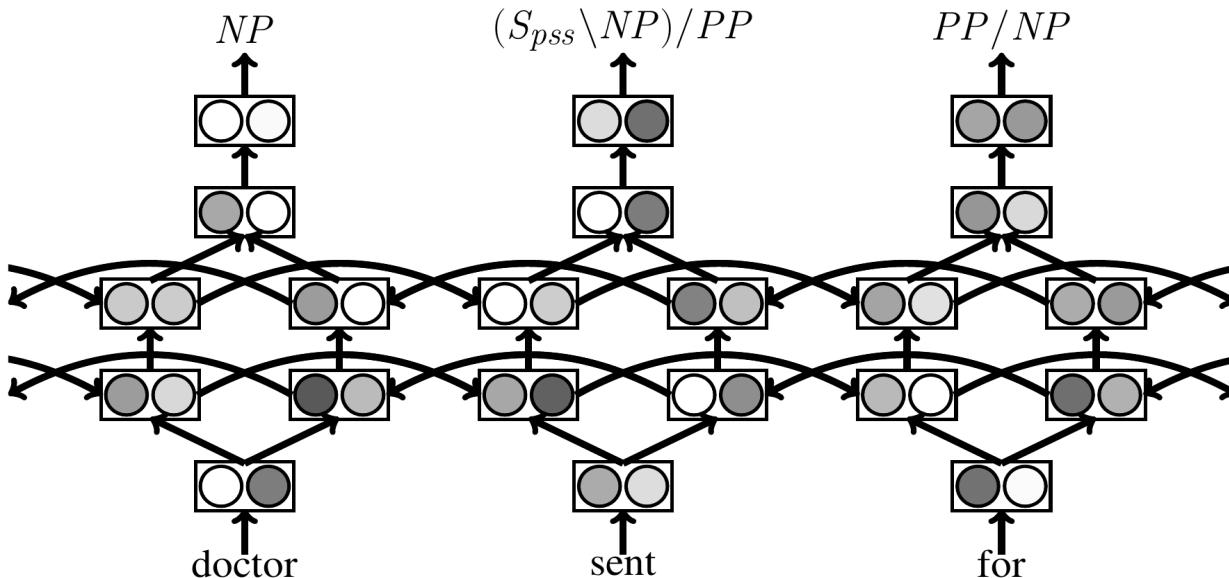
Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011.  
2016-10-14  
Natural Language Processing (Almost) from Scratch. J. Mach. Learn. Res. 12, 2493-2537.

# Results

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
<b>Benchmark Systems</b>	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40

- Window approach: POS, Chunking, NER
- Sentence approach: SRL
- WLL: Word-Level Log-Likelihood

# CCG Supertagging



Lewis, M., & Steedman, M. (2014). Improved CCG Parsing with Semi-supervised Supertagging. TACL.

Xu, W., Alipourfard, M., & Clark, S. (2015). CCG Supertagging with a Recurrent Neural Network. ACL.

Lewis, M., Lee, Kenton., & Zettlemoyer, L. (2016). LSTM CCG Parsing. NAACL.

# CCG Supertagging

- No POS feature
- Reduce word sparsity
- Global context information

Model	Accuracy	Time
C&C (gold POS)	92.60	-
C&C (auto POS)	91.50	0.57
NN	91.10	21.00
RNN	92.63	-
RNN+dropout	93.07	2.02

Model	Dev	Test
C&C tagger	91.5	92.0
NN	91.3	91.6
RNN	93.1	93.0
LSTM	94.1	94.3
LSTM + Tri-training	<b>94.9</b>	<b>94.7</b>

Lewis, M., & Steedman, M. (2014). Improved CCG Parsing with Semi-supervised Supertagging. TACL.

Xu, W., Alipour, M., & Clark, S. (2015). CCG Supertagging with a Recurrent Neural Network. ACL.

Lewis, M., Lee, Kenton., & Zettlemoyer, L. (2016). LSTM CCG Parsing. NAACL.

# CCG Supertagging

Supertagger	Accuracy
Bidirectional RNNs	93.4
Forward LSTM only	83.5
Backward LSTM only	89.5
Bidirectional LSTMs	<b>94.1</b>

Word Class	NN	LSTM	LSTM+ Tri-training
All	91.32	94.14	<b>94.90</b>
Unseen Words	90.39	94.21	<b>95.26</b>
Unseen Usages	45.80	59.37	<b>62.46</b>
Prepositions	78.11	84.40	<b>85.98</b>
Verbs	82.55	87.85	<b>89.24</b>
Wh-words	90.47	92.09	<b>94.16</b>
Long range	74.80	83.99	<b>86.31</b>

Lewis, M., & Steedman, M. (2014). Improved CCG Parsing with Semi-supervised Supertagging. TACL.

Xu, W., Alipourfard, M., & Clark, S. (2015). CCG Supertagging with a Recurrent Neural Network. ACL.

Lewis, M., Lee, Kenton., & Zettlemoyer, L. (2016). LSTM CCG Parsing. NAACL.

# CCG Supertagging

- Parsing results

Model	P	R	F1
C&C	86.2	84.2	85.2
C&C + RNN	87.7	86.4	87.0
EASYCCG	83.7	83.0	83.3
Dependencies	86.5	85.8	86.1
LSTM	87.7	86.7	87.2
LSTM + Dependencies	88.2	87.3	87.8
LSTM + Tri-training	<b>88.6</b>	<b>87.5</b>	<b>88.1</b>
LSTM + Tri-training + Dependencies	88.2	87.3	87.8

Lewis, M., & Steedman, M. (2014). Improved CCG Parsing with Semi-supervised Supertagging. TACL.

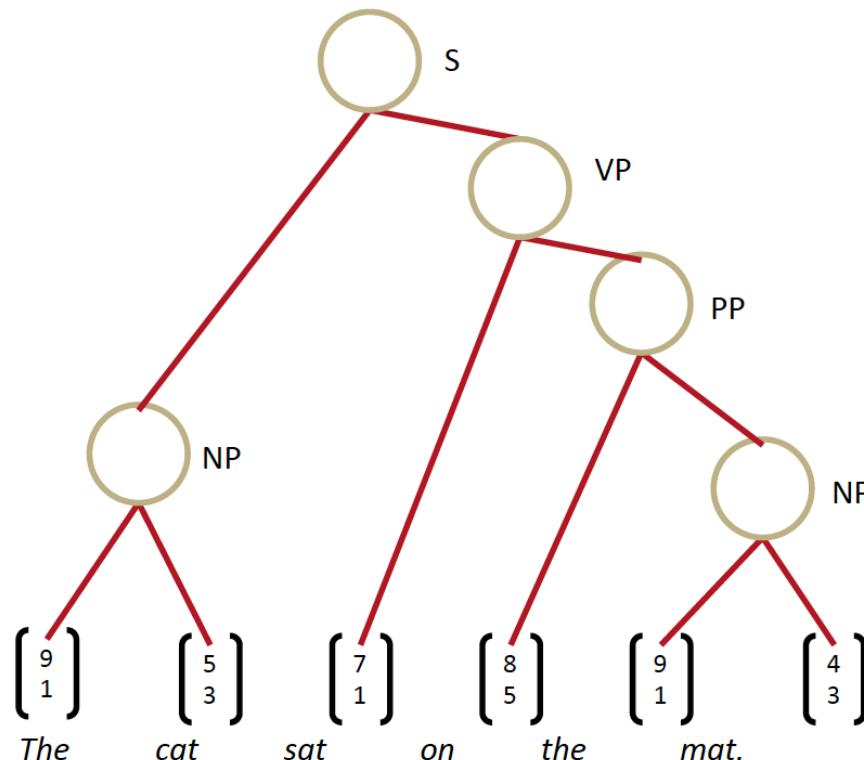
Xu, W., Alipourfard, M., & Clark, S. (2015). CCG Supertagging with a Recurrent Neural Network. ACL.

Lewis, M., Lee, Kenton., & Zettlemoyer, L. (2016). LSTM CCG Parsing. NAACL.

# Part 3.2: Greedy Search for Constituent Parsing with RNN

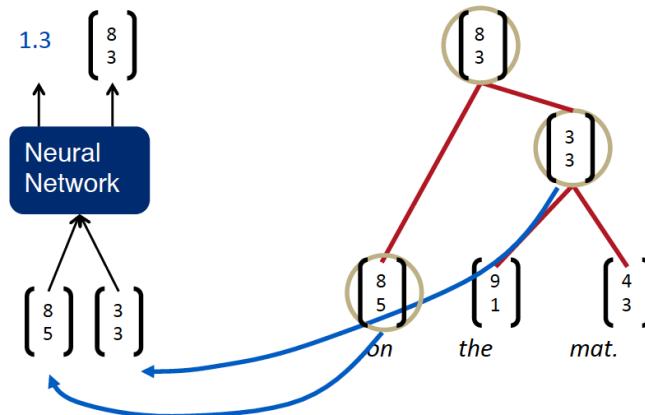
# Constituent Parsing with Recursive NN

- Our goal

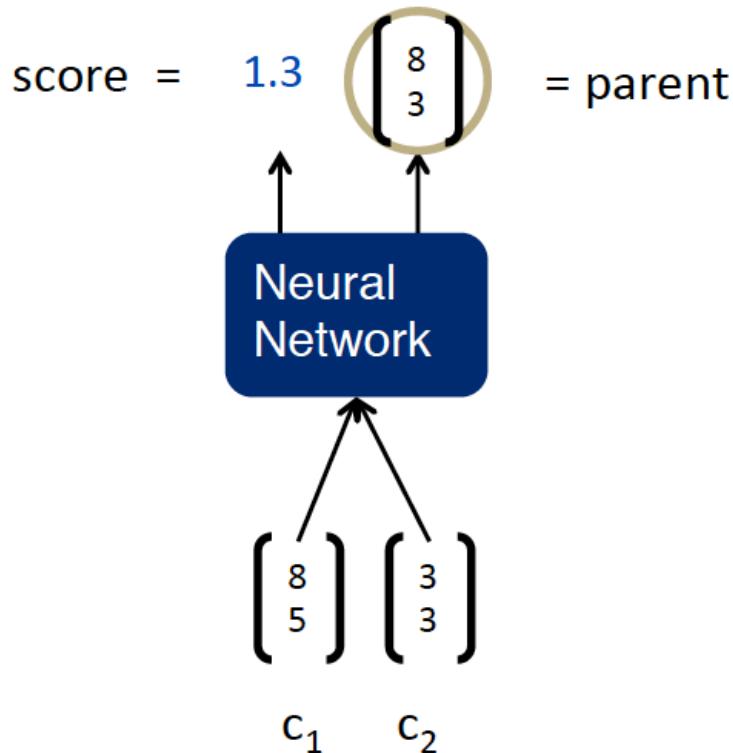


# Recursive NN

- Inputs
  - Two candidate children's representations
- Outputs
  - The semantic representation if the two nodes are merged
  - Score of how plausible the new node would be



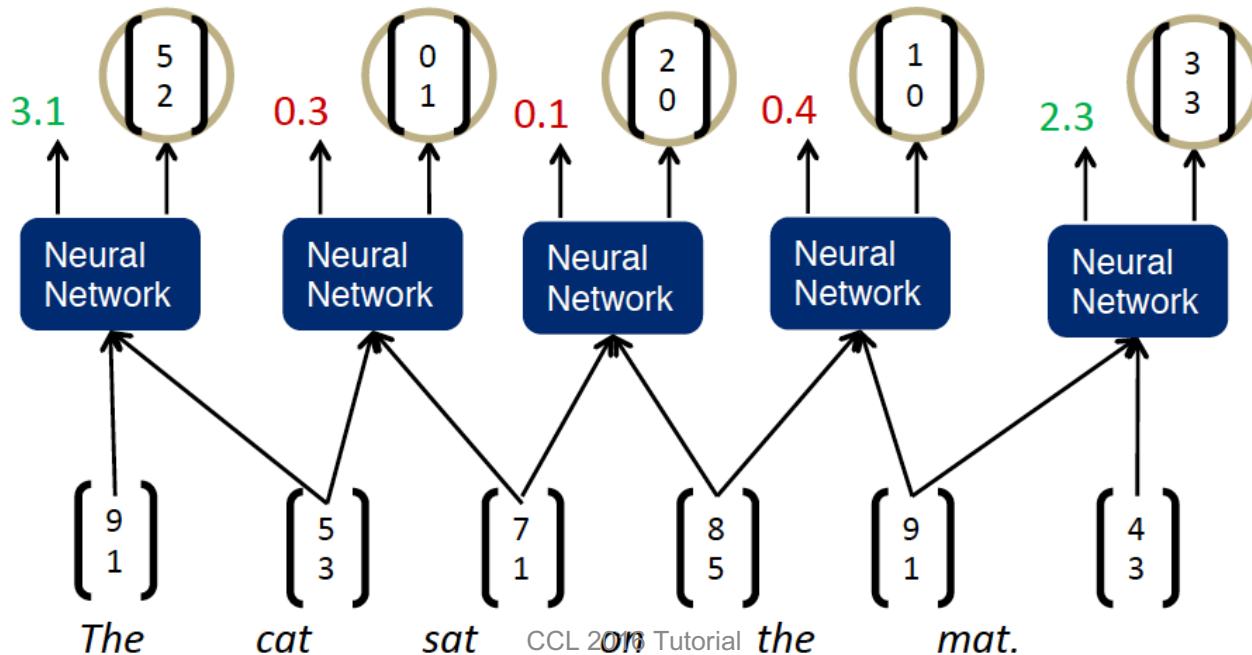
# RNN Definition



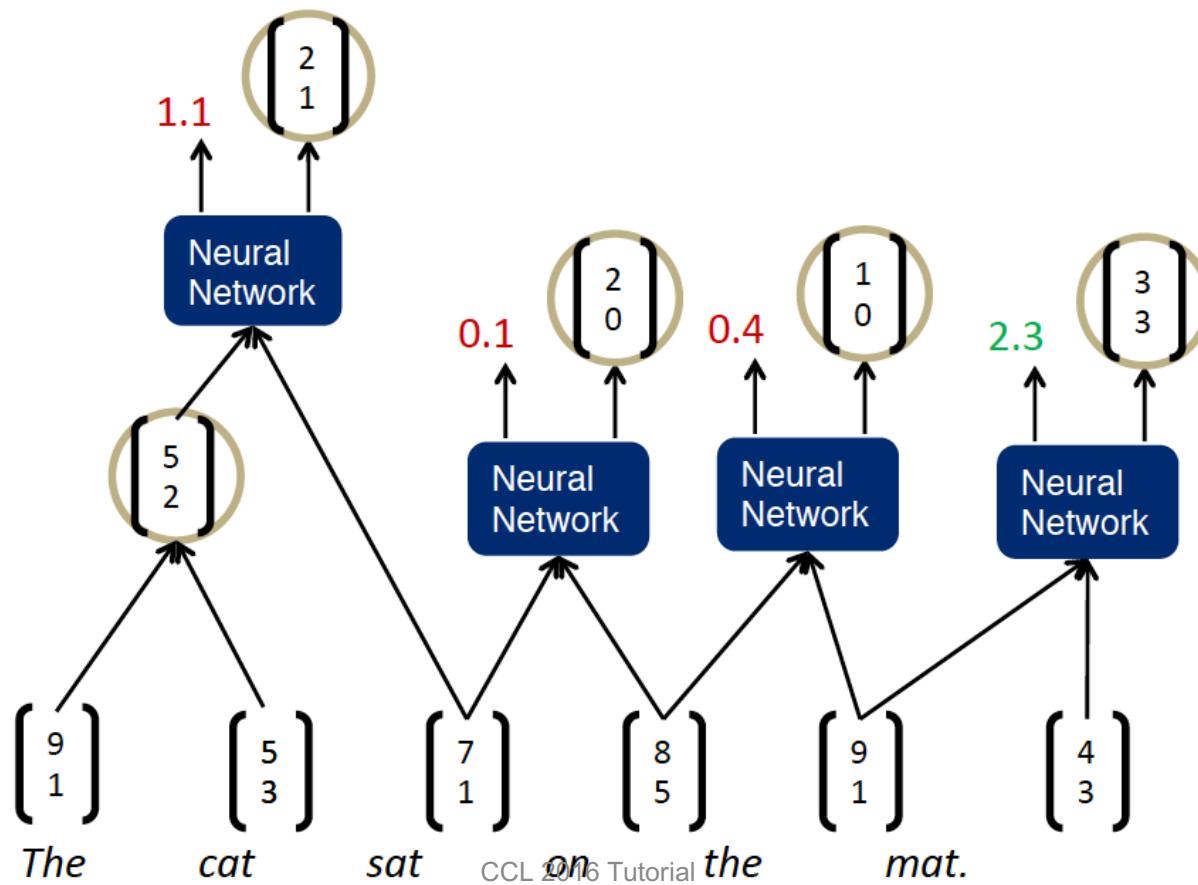
$$\left\{ \begin{array}{l} \text{score} = U^T p \\ p = \tanh(W \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + b) \end{array} \right.$$

where  $W$  at all nodes of the tree are the same

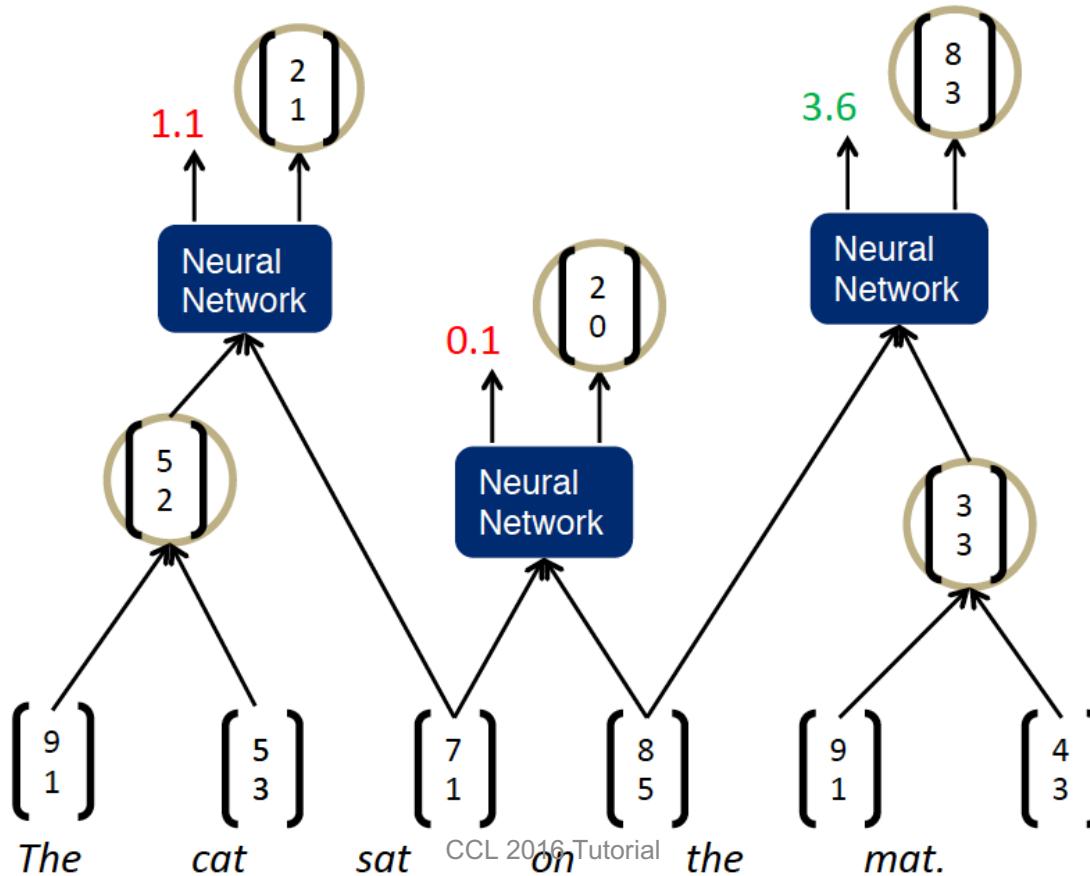
# Parsing a sentence with an RNN



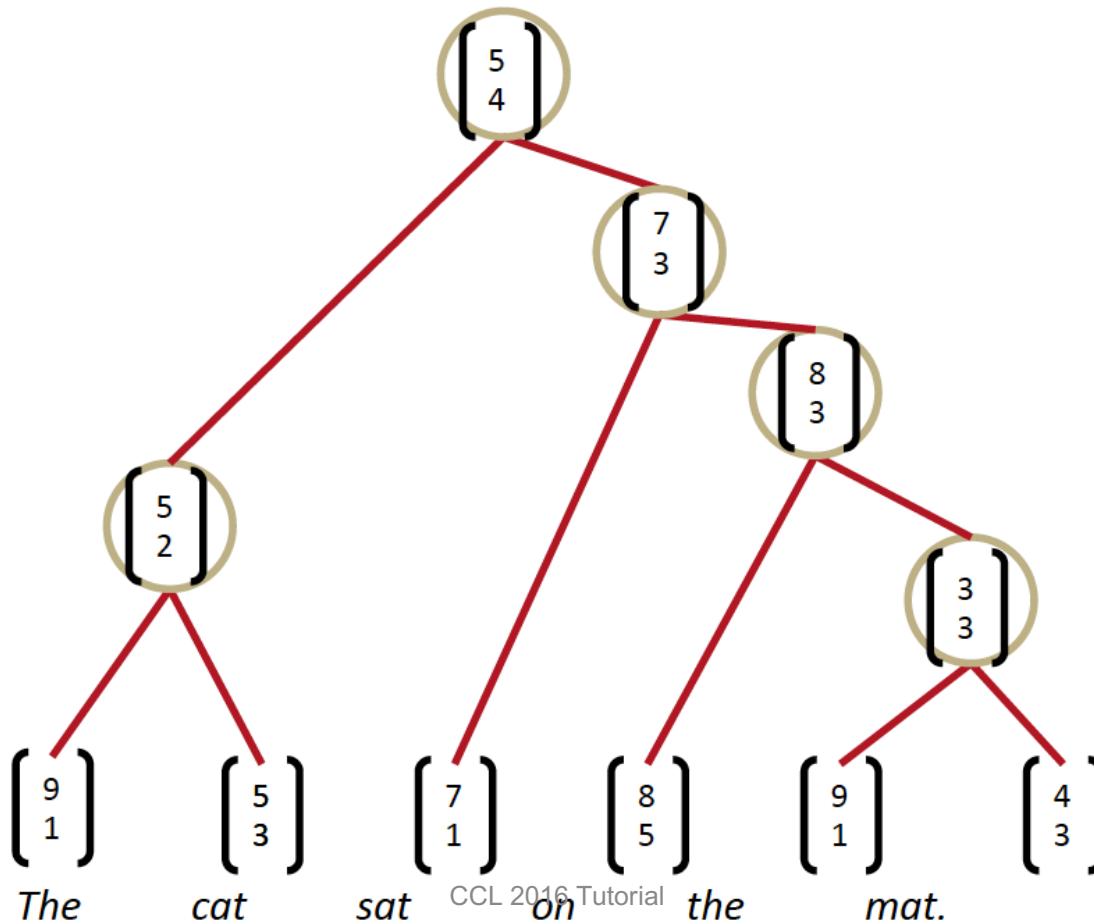
# Parsing a sentence with an RNN



# Parsing a sentence with an RNN

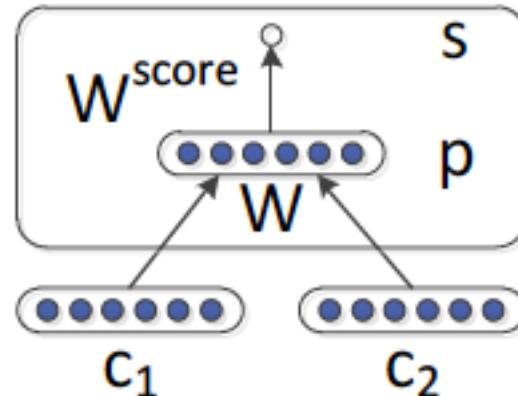


# Parsing a sentence with an RNN



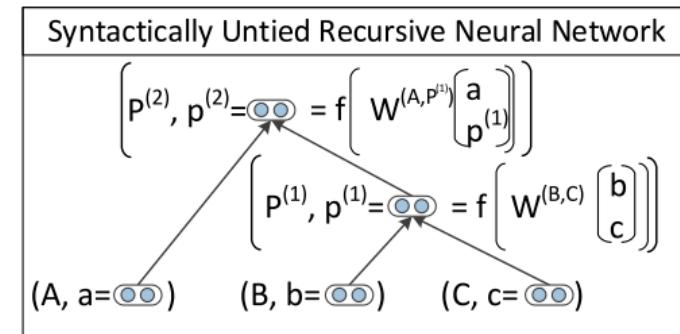
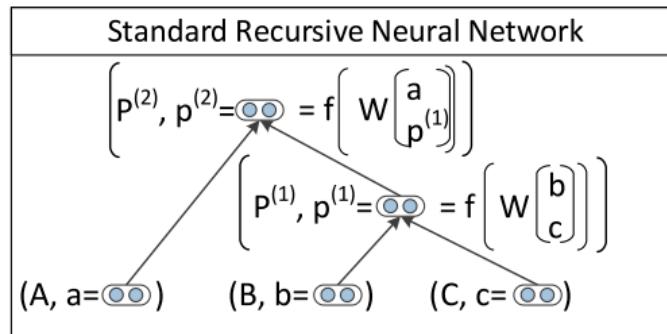
# Discussion on Simple RNN

- The composition function with single weight matrix is the same for all categories, punctuation, etc.
- It could capture some phenomena but not adequate for more complex, higher order composition



# Solution: Syntactically-Untied RNN (SU-RNN)

- Intuition
  - Condition the composition function on the syntactic categories
- Allows for different composition functions for pairs of syntactic categories, e.g. Adv + AdjP, VP + NP



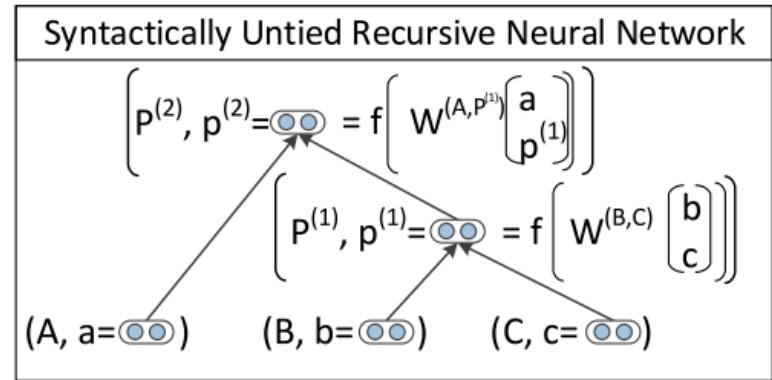
# Compositional Vector Grammars (CVG)

- PCFG + SU-RNN
- PCFG
  - Produce: k-best parsing trees
- SU-RNN
  - Re-ranking with SU-RNN

$$p^{(1)} = f \left( W^{(B,C)} \begin{bmatrix} b \\ c \end{bmatrix} \right)$$

$$s(p^{(1)}) = (v^{(B,C)})^T p^{(1)} + \log P(P_1 \rightarrow B \ C)$$

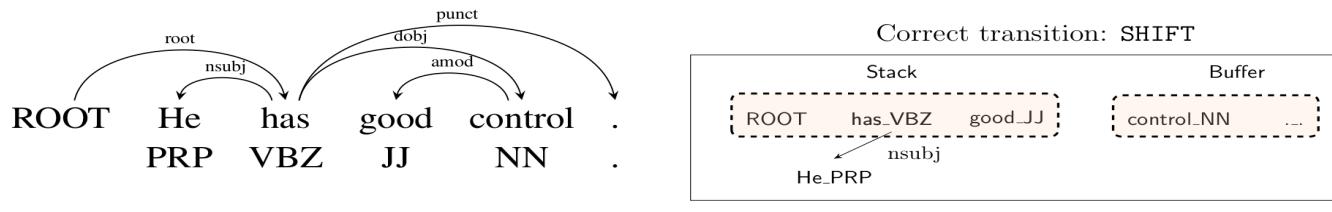
$$s(\text{CVG}(\theta, x, \hat{y})) = \sum_{d \in N(\hat{y})} s(p^d)$$



# Part 3.3: Transition-based Dependency Parsing with Greedy Search

# Dependency Parsing

- Neural MaltParser



Transition	Stack	Buffer	A
SHIFT	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
LEFT-ARC (nsubj)	[ROOT He has]	[good control .]	$A \cup \text{nsubj(has,He)}$
SHIFT	[ROOT has]	[good control .]	
SHIFT	[ROOT has good]	[control .]	
LEFT-ARC (amod)	[ROOT has good control]	[.]	$A \cup \text{amod(control,good)}$
RIGHT-ARC (dobj)	[ROOT has control]	[.]	$A \cup \text{dobj(has,control)}$
...	[ROOT has]	[.]	...
RIGHT-ARC (root)	[ROOT]	[.]	$A \cup \text{root(ROOT,has)}$

# Dependency Parsing

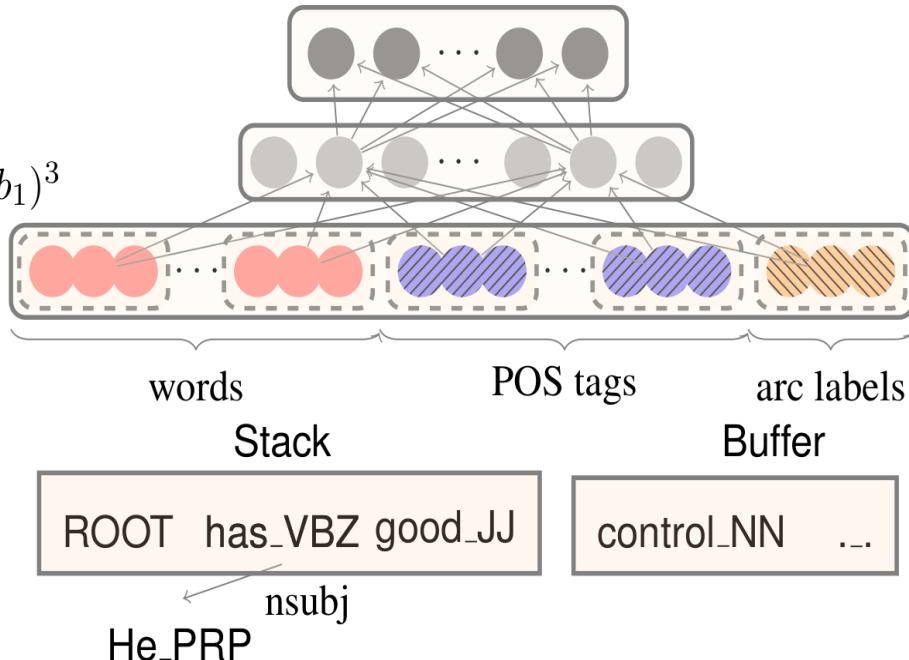
**Softmax layer:**

$$p = \text{softmax}(W_2 h)$$

**Hidden layer:**

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

**Input layer:**  $[x^w, x^t, x^l]$



# Dependency Parsing

- ZPar features (Zhang and Nivre, ACL 2011)

---

## Single-word features (9)

$s_1.w; s_1.t; s_1.wt; s_2.w; s_2.t;$   
 $s_2.wt; b_1.w; b_1.t; b_1.wt$

---

## Word-pair features (8)

$s_1.wt \circ s_2.wt; s_1.wt \circ s_2.w; s_1.wts_2.t;$   
 $s_1.w \circ s_2.wt; s_1.t \circ s_2.wt; s_1.w \circ s_2.w$   
 $s_1.t \circ s_2.t; s_1.t \circ b_1.t$

---

## Three-word feaures (8)

$s_2.t \circ s_1.t \circ b_1.t; s_2.t \circ s_1.t \circ lc_1(s_1).t;$   
 $s_2.t \circ s_1.t \circ rc_1(s_1).t; s_2.t \circ s_1.t \circ lc_1(s_2).t;$   
 $s_2.t \circ s_1.t \circ rc_1(s_2).t; s_2.t \circ s_1.w \circ rc_1(s_2).t;$   
 $s_2.t \circ s_1.w \circ lc_1(s_1).t; s_2.t \circ s_1.w \circ b_1.t$

---

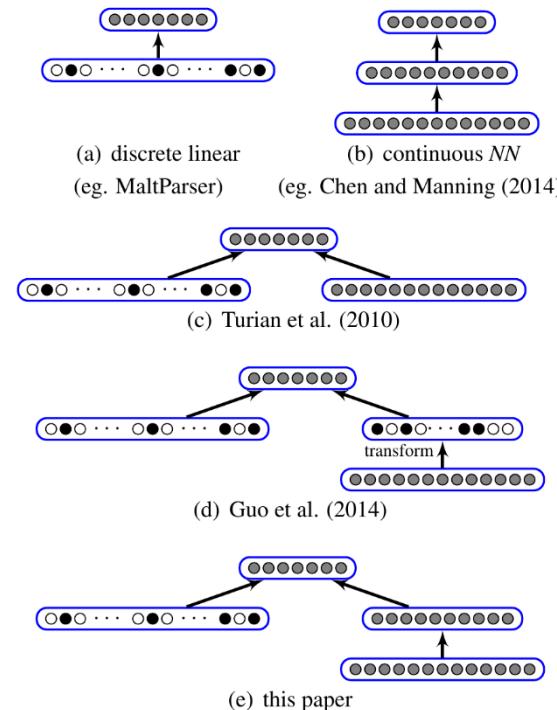
# Dependency Parsing

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	<b>92.0</b>	<b>89.7</b>	<b>91.8</b>	<b>89.6</b>	<b>654</b>

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	<b>84.0</b>	82.1	83.0	81.2	6
Our parser	<b>84.0</b>	<b>82.4</b>	<b>83.9</b>	<b>82.4</b>	<b>936</b>

# Dependency Parsing

- Chen and Manning with combined features



# Dependency Parsing

- Chen and Manning with combined features

System	UAS	LAS
<i>L</i>	89.36	88.33
<i>NN</i>	91.15	90.04
<i>This</i>	<b>91.80</b>	<b>90.68</b>
ZPar-local	89.94	88.92
Ma et al. (2014a)	90.38	–
Chen and Manning (2014)	91.17	89.99
Honnibal et al. (2013)	91.30	90.00
Ma et al. (2014a)*	91.32	–

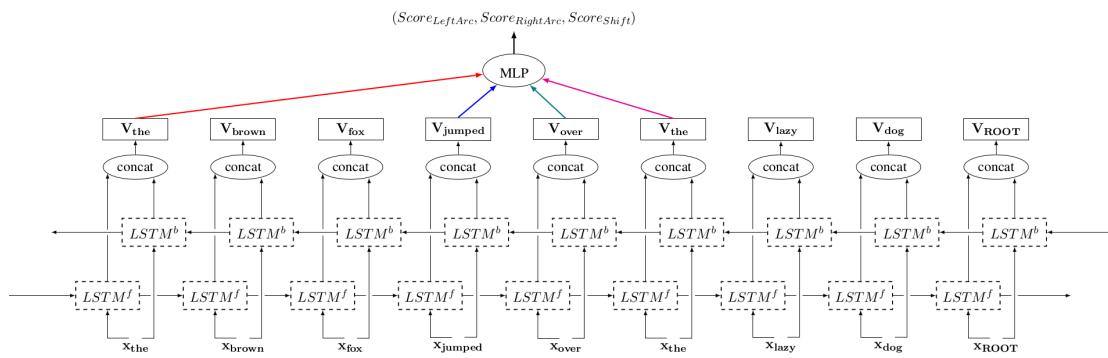
# Dependency Parsing

- Chen and Manning with richer features

Configuration:



Scoring:



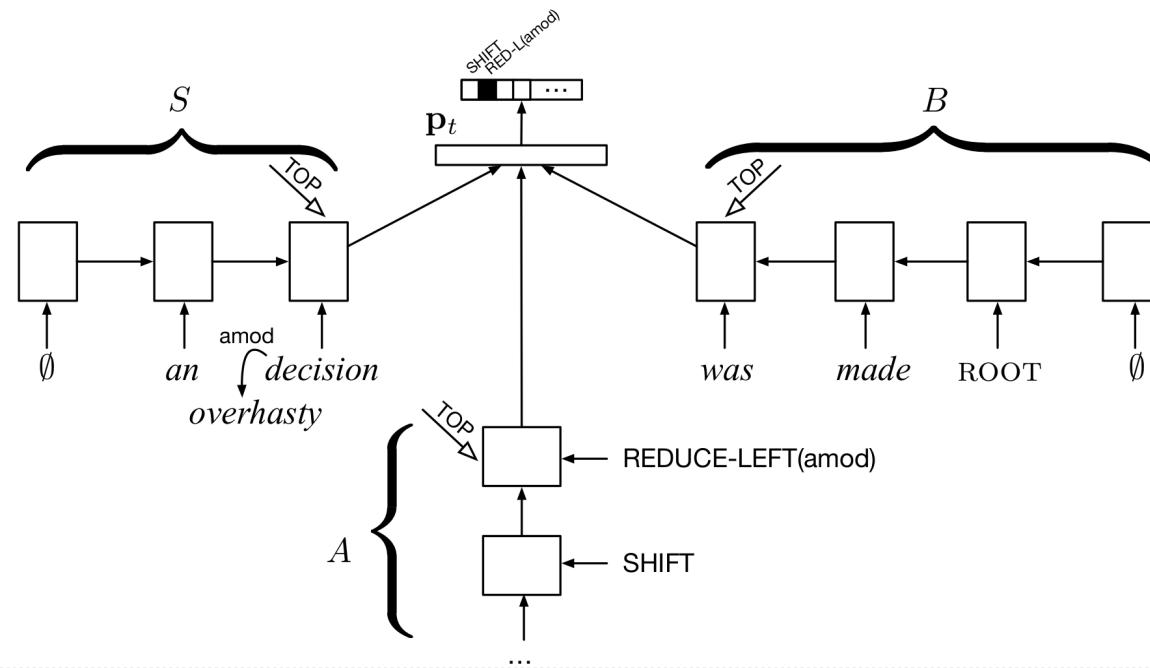
- 11 more LSTMs

# Dependency Parsing

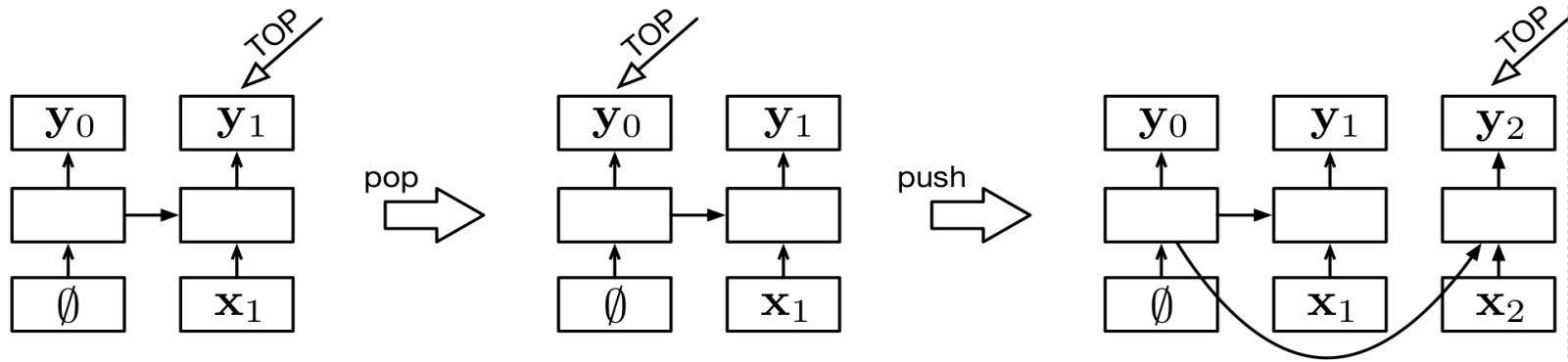
System	Method	Representation	Emb	PTB-YM		PTB-SD		CTB	
				UAS		UAS	LAS	UAS	LAS
This work	graph, 1st order	2 BiLSTM vectors	–	–	93.1	91.0	<b>86.6</b>	<b>85.1</b>	
This work	transition (greedy, dyn-oracle)	4 BiLSTM vectors	–	–	93.1	91.0	86.2	85.0	
This work	transition (greedy, dyn-oracle)	11 BiLSTM vectors	–	–	<b>93.2</b>	<b>91.2</b>	86.5	84.9	
ZhangNivre11	transition (beam)	large feature set (sparse)	–	92.9	–	–	86.0	84.4	
Martins13 (TurboParser)	graph, 3rd order+	large feature set (sparse)	–	92.8	93.1	–	–	–	–
Pei15	graph, 2nd order	large feature set (dense)	–	93.0	–	–	–	–	–
Dyer15	transition (greedy)	Stack-LSTM + composition	–	–	92.4	90.0	85.7	84.1	
Ballesteros16	transition (greedy, dyn-oracle)	Stack-LSTM + composition	–	–	92.7	90.6	86.1	84.5	
This work	graph, 1st order	2 BiLSTM vectors	YES	–	93.0	90.9	86.5	84.9	
This work	transition (greedy, dyn-oracle)	4 BiLSTM vectors	YES	–	93.6	91.5	87.4	85.9	
This work	transition (greedy, dyn-oracle)	11 BiLSTM vectors	YES	–	93.9	91.9	<b>87.6</b>	86.1	
Weiss15	transition (greedy)	large feature set (dense)	YES	–	93.2	91.2	–	–	
Weiss15	transition (beam)	large feature set (dense)	YES	–	<b>94.0</b>	<b>92.0</b>	–	–	
Pei15	graph, 2nd order	large feature set (dense)	YES	93.3	–	–	–	–	–
Dyer15	transition (greedy)	Stack-LSTM + composition	YES	–	93.1	90.9	87.1	85.5	
Ballesteros16	transition (greedy, dyn-oracle)	Stack-LSTM + composition	YES	–	93.6	91.4	<b>87.6</b>	<b>86.2</b>	
LeZuidema14	reranking /blend	inside-outside recursive net	YES	93.1	93.8	91.5	–	–	
Zhu15	reranking /blend	recursive conv-net	YES	93.8	–	–	85.7	–	

# Dependency Parsing

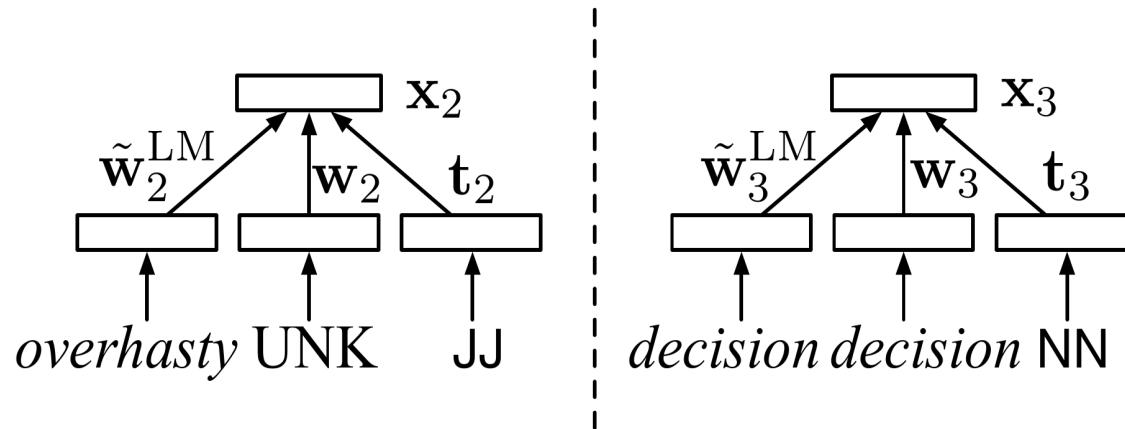
- Chen and Manning with less features



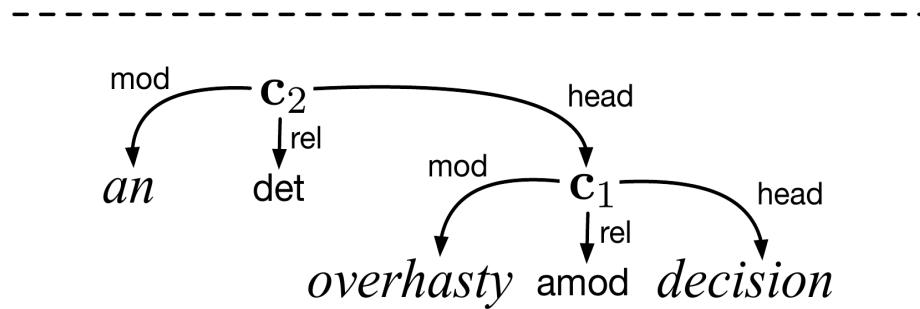
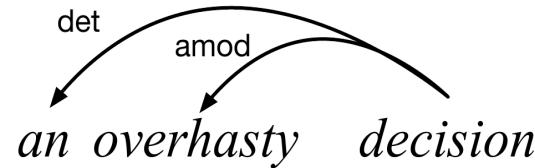
# Dependency Parsing



# Dependency Parsing



# Dependency Parsing



# Dependency Parsing

	Development		Test	
	UAS	LAS	UAS	LAS
S-LSTM	<b>93.2</b>	<b>90.9</b>	<b>93.1</b>	<b>90.9</b>
-POS	93.1	90.4	92.7	90.3
-pretraining	92.7	90.4	92.4	90.0
-composition	92.7	89.9	92.2	89.6
S-RNN	92.8	90.4	92.3	90.1
C&M (2014)	92.2	89.7	91.8	89.6

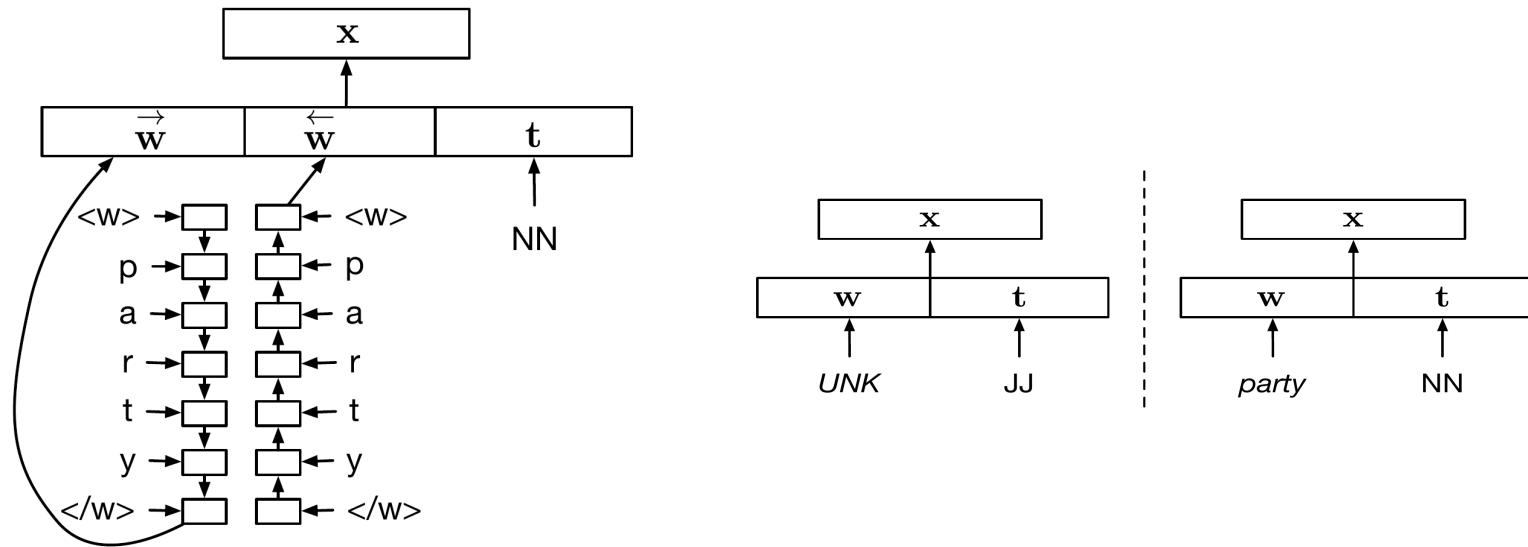
English parsing results (SD)

	Development		Test	
	UAS	LAS	UAS	LAS
S-LSTM	<b>87.2</b>	<b>85.9</b>	<b>87.2</b>	<b>85.7</b>
-POS	82.8	79.8	82.2	79.1
-pretraining	86.3	84.7	85.7	84.1
-composition	85.8	84.0	85.3	83.6
S-RNN	86.3	84.7	86.1	84.6
C&M (2014)	84.0	82.4	83.9	82.4

Chinese parsing results (CTB5)

# Dependency Parsing

- Dyer et al. with character based word vector



# Dependency Parsing

UAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	86.14	<b>87.20</b>	<b>87.44</b>	87.07
Basque	78.42	<b>84.97</b>	83.49	<b>85.58</b>
French	84.84	<b>86.21</b>	<b>87.00</b>	86.33
German	88.14	<b>90.94</b>	91.16	<b>91.23</b>
Hebrew	79.73	<b>79.92</b>	<b>81.99</b>	80.76
Hungarian	72.38	<b>80.16</b>	78.47	<b>80.85</b>
Korean	78.98	<b>88.98</b>	87.36	<b>89.14</b>
Polish	73.29	<b>85.69</b>	<b>89.32</b>	88.54
Swedish	73.44	<b>75.03</b>	<b>80.02</b>	78.85
Turkish	71.10	<b>74.91</b>	77.13	<b>77.96</b>
Chinese	79.43	<b>80.36</b>	<b>85.98</b>	85.81
English	91.64	<b>91.98</b>	<b>92.94</b>	92.49
Average	79.79	<b>83.86</b>	85.19	<b>85.38</b>

LAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	82.73	<b>84.34</b>	<b>84.81</b>	84.36
Basque	67.08	<b>78.22</b>	74.31	<b>79.52</b>
French	80.32	<b>81.70</b>	<b>82.71</b>	81.51
German	85.36	<b>88.68</b>	<b>89.04</b>	88.83
Hebrew	69.42	<b>70.58</b>	<b>74.11</b>	72.18
Hungarian	62.14	<b>75.61</b>	69.50	<b>76.16</b>
Korean	67.48	<b>86.80</b>	83.80	<b>86.88</b>
Polish	65.13	<b>78.23</b>	<b>81.84</b>	80.97
Swedish	64.77	<b>66.74</b>	<b>72.09</b>	69.88
Turkish	53.98	<b>62.91</b>	62.30	<b>62.87</b>
Chinese	75.64	<b>77.06</b>	<b>84.36</b>	84.10
English	88.60	<b>89.58</b>	<b>90.63</b>	90.08
Average	71.89	<b>78.37</b>	79.13	<b>79.78</b>

# Dependency Parsing

UAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	85.21	<b>86.08</b>	86.05	<b>86.07</b>
Basque	77.06	<b>85.19</b>	82.92	<b>85.22</b>
French	83.74	<b>85.34</b>	<b>86.15</b>	85.78
German	82.75	<b>86.80</b>	<b>87.33</b>	87.26
Hebrew	77.62	<b>79.93</b>	<b>80.68</b>	80.17
Hungarian	72.78	<b>80.35</b>	78.64	<b>80.92</b>
Korean	78.70	<b>88.39</b>	86.85	<b>88.30</b>
Polish	72.01	<b>83.44</b>	<b>87.06</b>	85.97
Swedish	76.39	<b>79.18</b>	<b>83.43</b>	83.24
Turkish	71.70	<b>76.32</b>	75.32	<b>76.34</b>
Chinese	79.01	<b>79.94</b>	<b>85.96</b>	85.30
English	91.16	<b>91.47</b>	<b>92.57</b>	91.63
Average	79.01	<b>85.36</b>	84.41	<b>84.68</b>

LAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	82.05	<b>83.41</b>	<b>83.46</b>	83.40
Basque	66.61	<b>79.09</b>	73.56	<b>78.61</b>
French	79.22	<b>80.92</b>	<b>82.03</b>	81.08
German	79.15	<b>84.04</b>	<b>84.62</b>	84.49
Hebrew	68.71	<b>71.26</b>	<b>72.70</b>	72.26
Hungarian	61.93	<b>75.19</b>	69.31	<b>76.34</b>
Korean	67.50	<b>86.27</b>	83.37	<b>86.21</b>
Polish	63.96	<b>76.84</b>	<b>79.83</b>	78.24
Swedish	67.69	<b>71.19</b>	<b>76.40</b>	74.47
Turkish	54.55	<b>64.34</b>	61.22	<b>62.28</b>
Chinese	74.79	<b>76.29</b>	<b>84.40</b>	83.72
English	88.42	<b>88.94</b>	<b>90.31</b>	89.44
Average	71.22	<b>78.15</b>	78.43	<b>79.21</b>

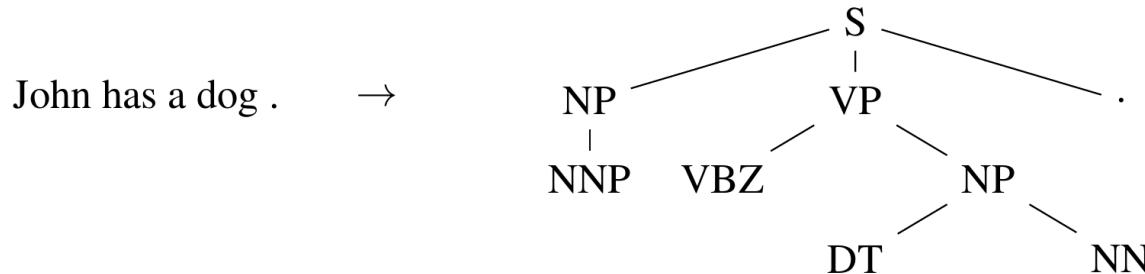
# Dependency Parsing

Language	This Work			Best Greedy Result			Best Published Result		
	UAS	LAS	System	UAS	LAS	System	UAS	LAS	System
Arabic	86.08	83.41	Chars	84.57	81.90	B'13	88.32	86.21	B+'13
Basque	85.22	78.61	Chars + POS	84.33	78.58	B'13	89.96	85.70	B+'14
French	86.15	82.03	Words + POS	83.35	77.98	B'13	89.02	85.66	B+'14
German	87.33	84.62	Words + POS	85.38	82.75	B'13	91.64	89.65	B+'13
Hebrew	80.68	72.70	Words + POS	79.89	73.01	B'13	87.41	81.65	B+'14
Hungarian	80.92	76.34	Chars + POS	83.71	79.63	B'13	89.81	86.13	B+'13
Korean	88.39	86.27	Chars	85.72	82.06	B'13	89.10	87.27	B+'14
Polish	87.06	79.83	Words + POS	85.80	79.89	B'13	91.75	87.07	B+'13
Swedish	83.43	76.40	Words + POS	83.20	75.82	B'13	88.48	82.75	B+'14
Turkish	76.32	64.34	Chars	75.82	65.68	N+'06a	77.55	n/a	K+'10
Chinese	85.96	84.40	Words + POS	87.20	85.70	D+'15	87.20	85.70	D+'15
English	92.57	90.31	Words + POS	93.10	90.90	D+'15	94.08	92.19	W+'15

# Part 3.4: Sequence to Sequence with Greedy Search

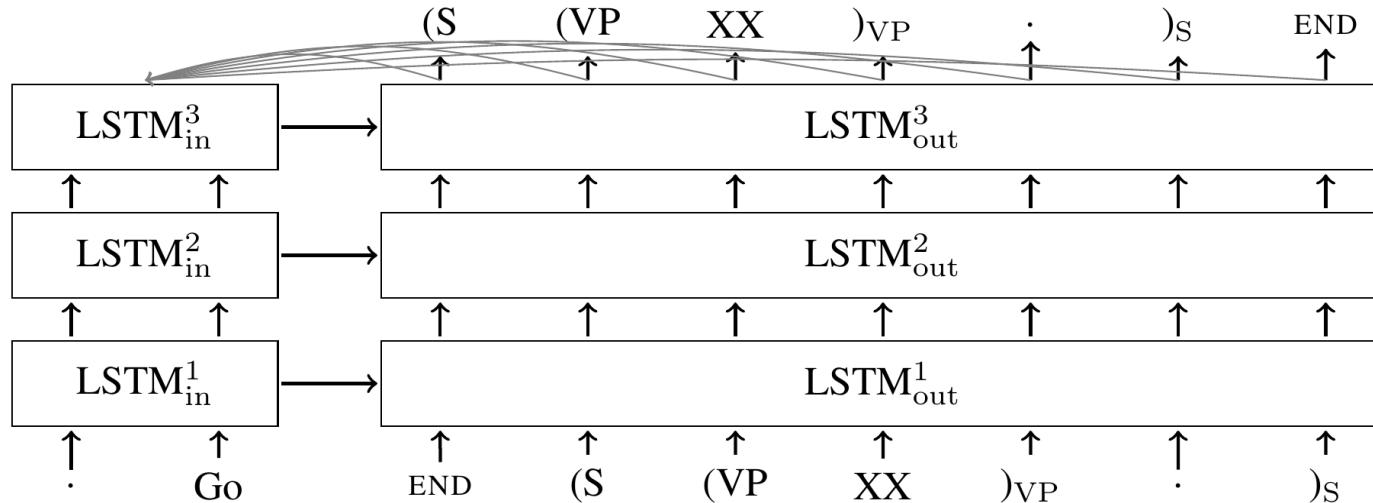
# Constituent Parsing

- Sequence to sequence



John has a dog . → (S (NP NNP )<sub>NP</sub> (VP VBZ (NP DT NN )<sub>NP</sub> )<sub>VP</sub> . )<sub>S</sub>

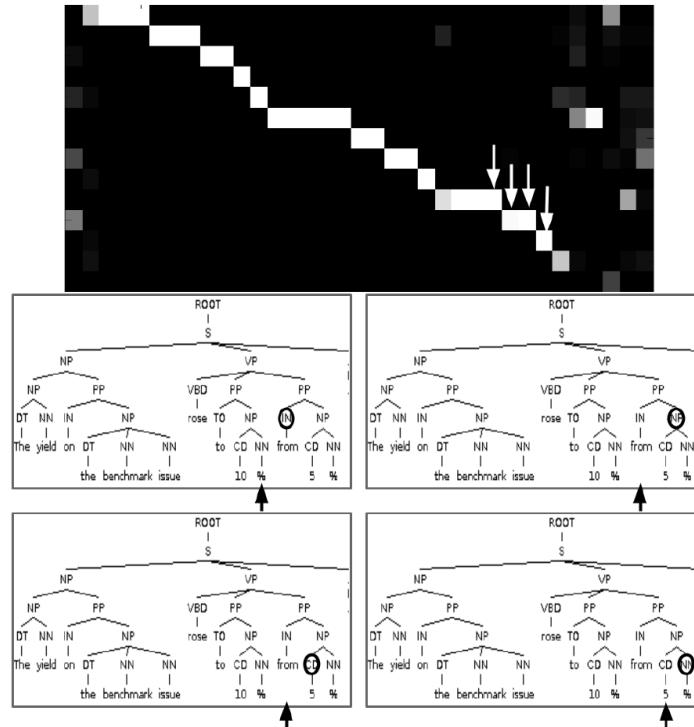
# Constituent Parsing



# Constituent Parsing

Parser	Training Set	WSJ 22	WSJ 23
baseline LSTM+D LSTM+A+D LSTM+A+D ensemble	WSJ only	< 70	< 70
	WSJ only	88.7	88.3
	WSJ only	90.7	90.5
baseline LSTM LSTM+A	BerkeleyParser corpus	91.0	90.5
	high-confidence corpus	<b>92.8</b>	<b>92.1</b>
Petrov et al. (2006) [12] Zhu et al. (2013) [13] Petrov et al. (2010) ensemble [14]	WSJ only	91.1	90.4
	WSJ only	N/A	90.4
	WSJ only	92.5	91.8
Zhu et al. (2013) [13] Huang & Harper (2009) [15] McClosky et al. (2006) [16]	semi-supervised	N/A	91.3
	semi-supervised	N/A	91.3
	semi-supervised	92.4	<b>92.1</b>

# Constituent Parsing



# Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che (HIT)

Yue Zhang (SUTD)

# Part 4: Dynamic Programming Decoding

# Part 4.1: Dynamic Programming Decoding for Tagging

# Word-Level Log-Likelihood

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
<b>Benchmark Systems</b>	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40

- WLL: Word-Level Log-Likelihood
  - Each word in a sentence is considered independently

# Sentence-Level Log-Likelihood

- Considering dependencies between tags in a sentence
- Conditional likelihood by **normalizing** all possible paths (CRF)
- Sentence score for one tag path

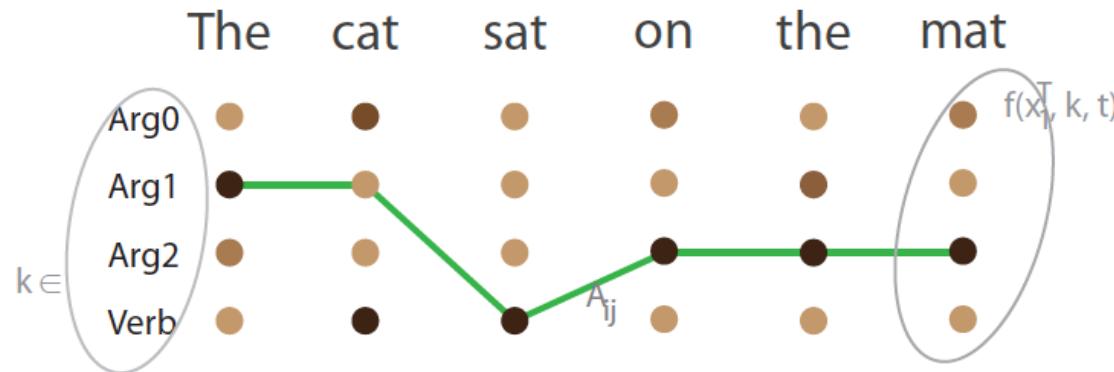
$$\log p([y]_1^T \mid [x]_1^T, \tilde{\theta}) = s([x]_1^T, [y]_1^T, \tilde{\theta}) - \text{logadd} s([x]_1^T, [j]_1^T, \tilde{\theta}) \\ \forall [j]_1^T$$

– where  $A_{[i][j]}$  is a transition score for jumping from tag  $i$  to  $j$

$$s([x]_1^T, [i]_1^T, \tilde{\theta}) = \sum_{t=1}^T \left( A_{[i]_{t-1}[i]_t} + f([x]_1^T, [i]_t, t, \theta) \right)$$

# Sentence-Level Log-Likelihood

- Decoding: finding the max scored path
  - Viterbi algorithm



# Results

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
<b>Benchmark Systems</b>	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99

- SLL helps, but fair performance for POS

# Improvements

- Supervised word embeddings

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
<b>Benchmark Systems</b>	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

- More (embedding) features

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL
<b>Benchmark Systems</b>	97.24	94.29	89.31	77.92
NN+SLL+LM2	97.20	93.63	88.67	74.15
NN+SLL+LM2+Suffix2	97.29	—	—	—
NN+SLL+LM2+Gazetteer	—	—	89.59	—
NN+SLL+LM2+POS	—	94.32	88.67	—
NN+SLL+LM2+CHUNK	—	—	—	74.72

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011.

Natural Language Processing (Almost) from Scratch. J. Mach. Learn. Res. 12 (November 2011), 2493-2537.

# Speed

System	RAM (Mb)	Time (s)
Toutanova, 2003	1100	1065
Shen, 2007	2200	833
SENNNA	32	4

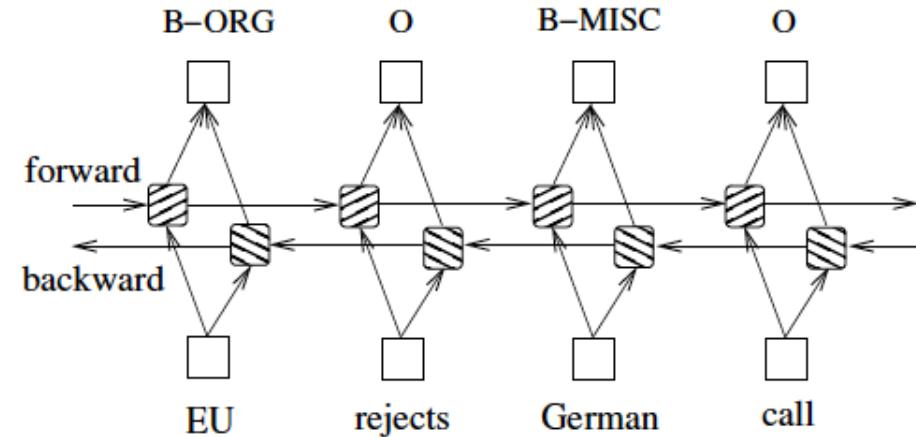
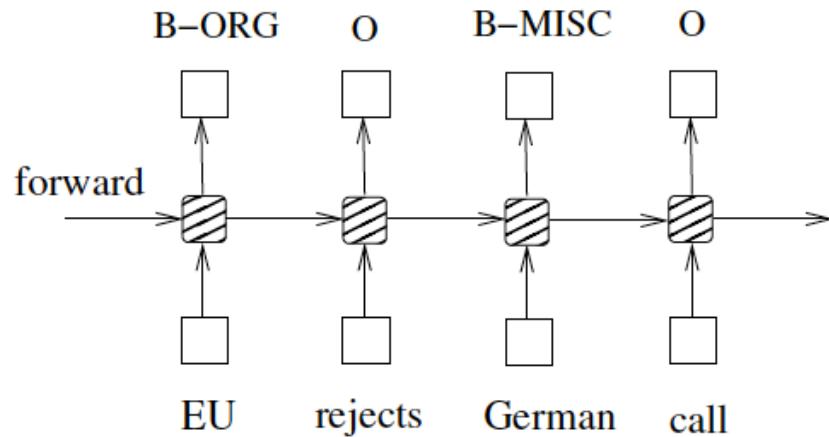
(a) POS

System	RAM (Mb)	Time (s)
Koomen, 2005	3400	6253
SENNNA	124	52

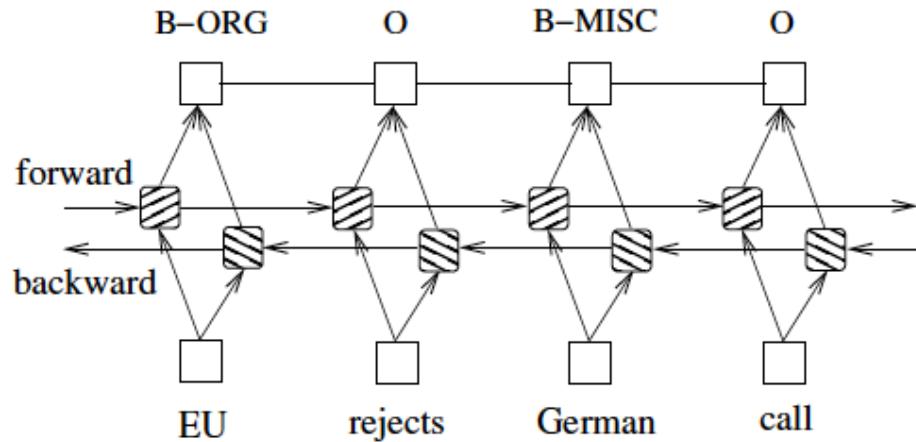
(b) SRL

# RNNs for Tagging

- LSTM
- Bi-LSTM



# Bi-LSTM-CRF



---

**Algorithm 1** Bidirectional LSTM CRF model training procedure

---

```
1: for each epoch do
2:   for each batch do
3:     1) bidirectional LSTM-CRF model forward pass:
4:     forward pass for forward state LSTM
5:     forward pass for backward state LSTM
6:     2) CRF layer forward and backward pass
7:     3) bidirectional LSTM-CRF model backward pass:
8:       backward pass for forward state LSTM
9:       backward pass for backward state LSTM
10:      4) update parameters
11:    end for
12: end for
```

---

# Results

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	<b>97.45</b>	93.80	84.10
	BI-LSTM-CRF	97.43	<b>94.13</b>	<b>84.26</b>
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	<b>97.55</b>	<b>94.46</b>	<b>88.83 (90.10)</b>

# BI-LSTM-CRF for SRL

- End-to-end tagging model
  - 8 layer bi-directional LSTM
  - No parsing features
- Features
  - Argument
  - Predicate
  - Predicate-context
  - Region-mark
- Achieving new SOTA

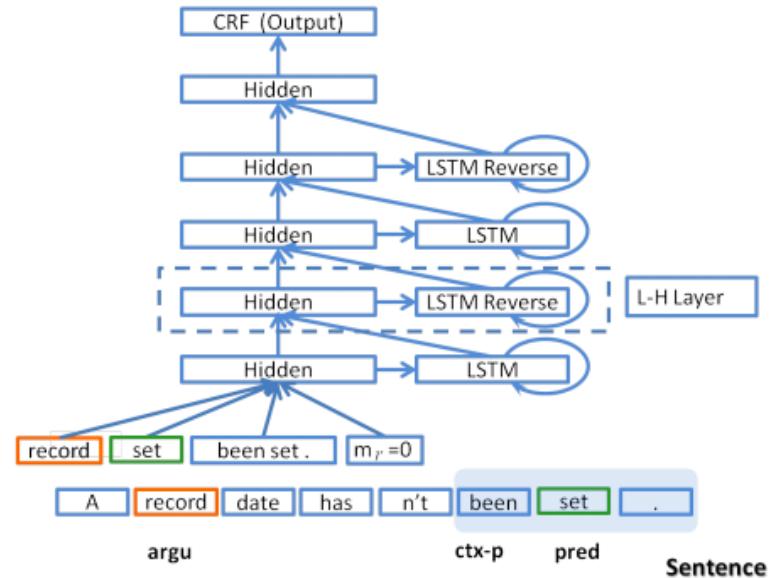
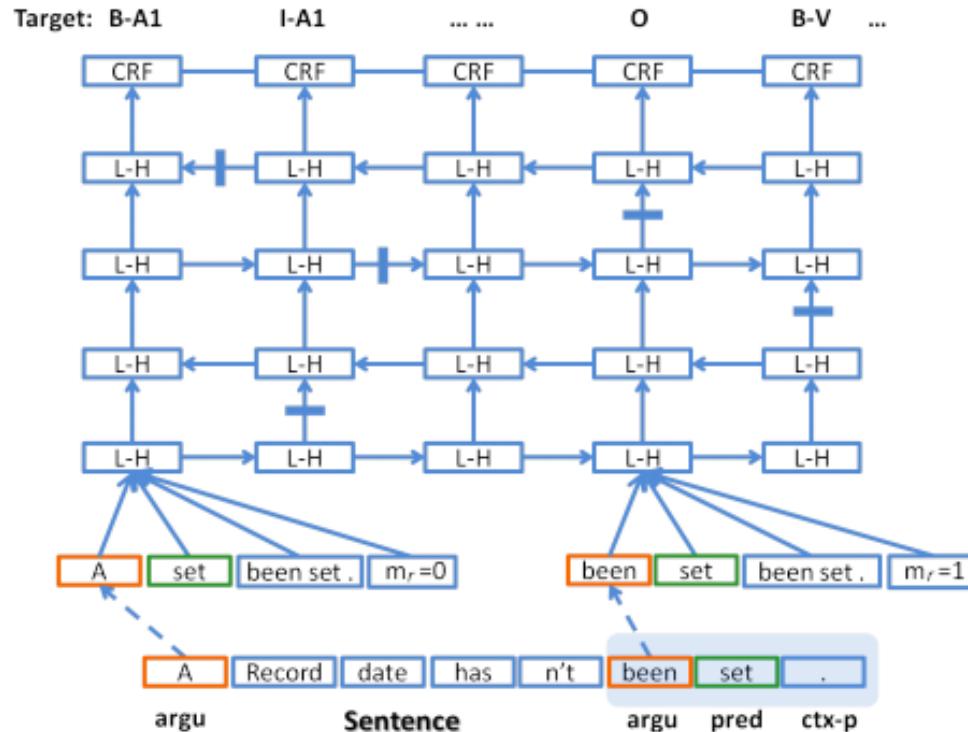
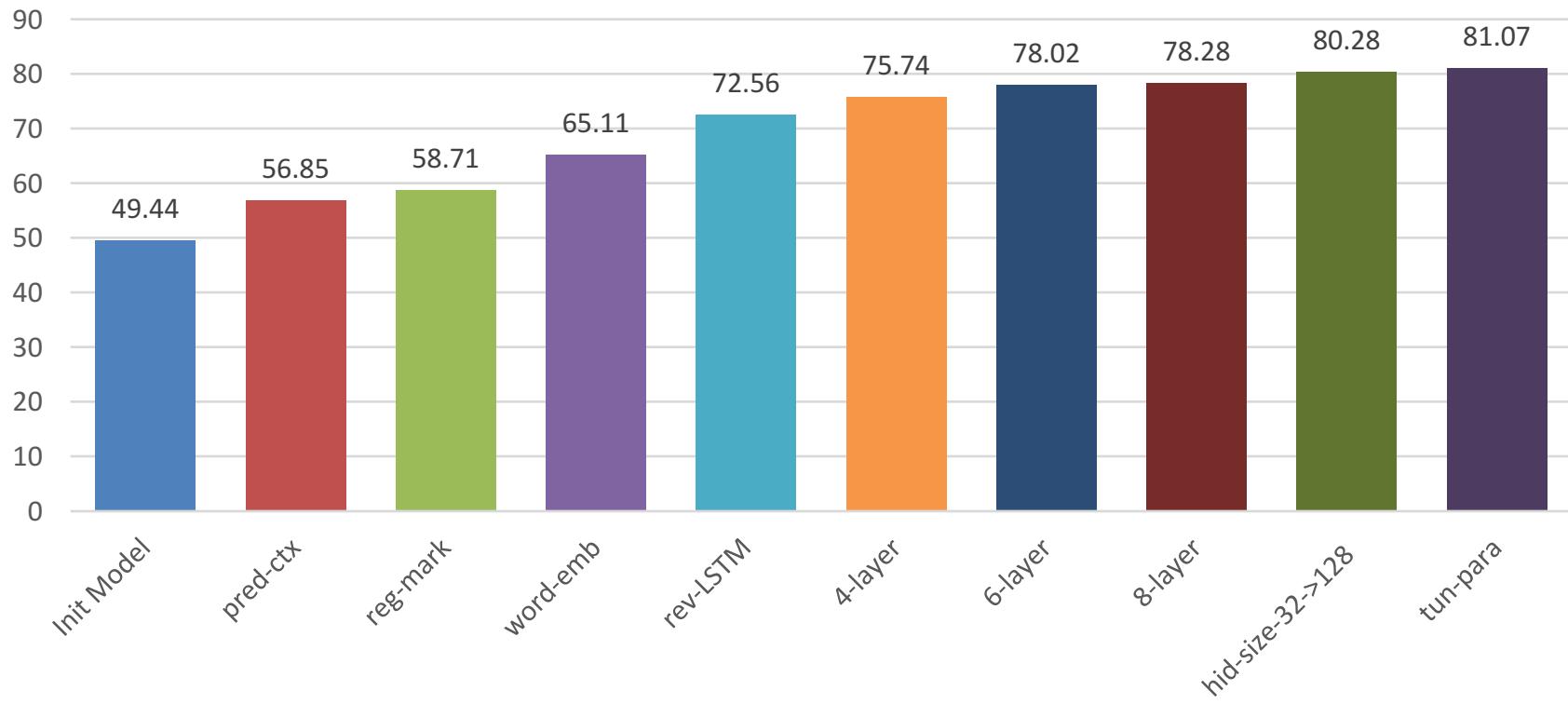


Figure 2: DB-LSTM network. Shadow part denote the predicate context within length 1.

# Temporal Expanded



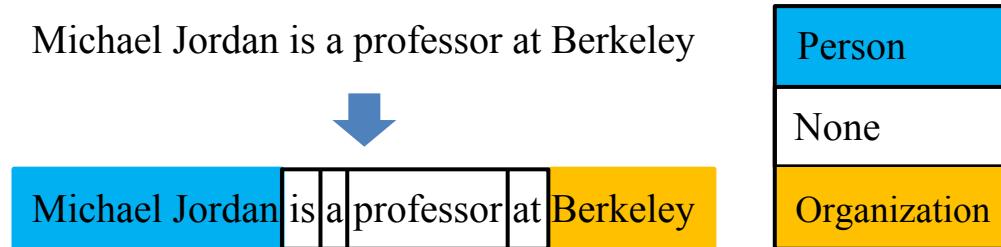
# Results



Jie Zhou and Wei Xu. (2015). End-to-end learning of semantic role labeling using recurrent neural networks. ACL.  
2016-10-14

# Segmentation Models

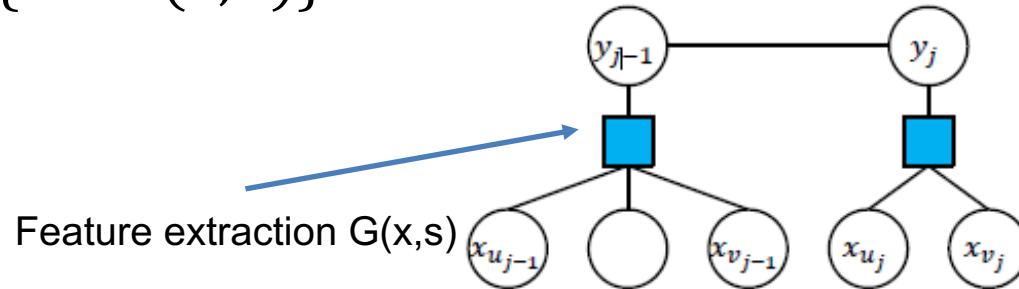
- Tagging models cannot extract segment information
  - E.g. the length of a segment
- Some tagging problems can be naturally modeled into segmentation task
  - E.g. word segmentation, named entity recognition



浦东开发与建设 → 浦东 / 开发 / 与 / 建设  
Pudong development and construction

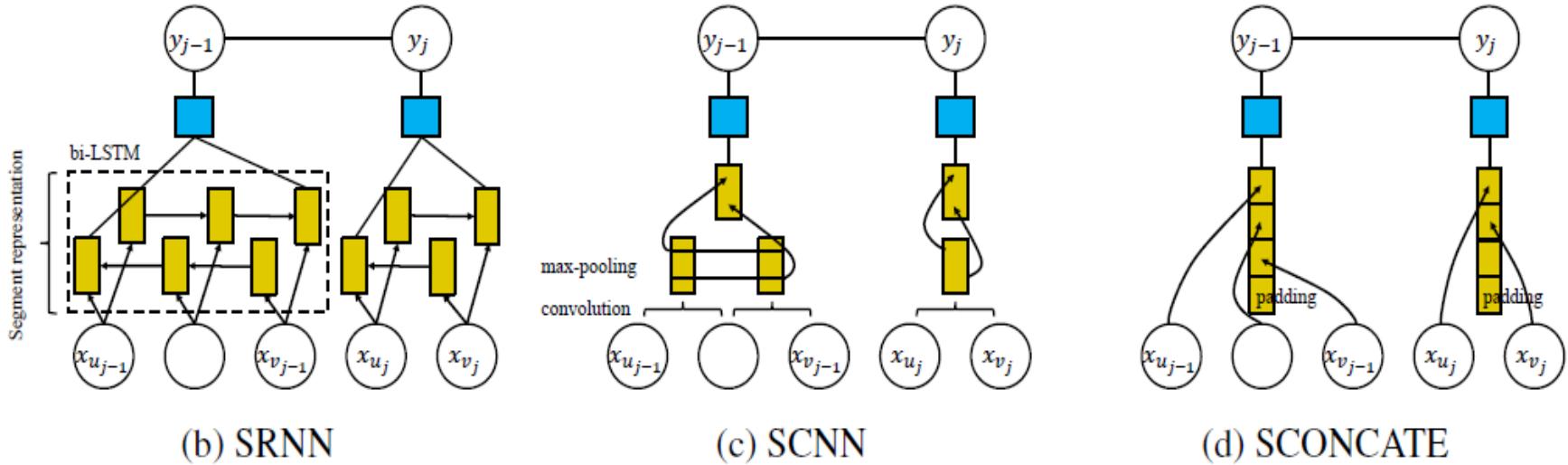
# Semi-CRF

- A solution
  - Semi-Markov CRF [Sarawagi and Cohen, 2004]
  - Modeling segments directly
  - $p(s|x) = \frac{1}{Z(x)} \exp\{W \cdot G(x, s)\}$



Can we represent segments with vectors?

# Compositional Segment Representation



# Decoding Algorithm

**Input:** a sequence  $X = (x_0, \dots, x_{n-1})$  of  $n$  units, the maximum length of the segment  $L$

**Output:** the highest scored segmentation  $S = (s_0, \dots, s_{m-1})$ , where  $s = (u, v, y)$  is a segment and  $u$  represents the starting position,  $v$  represents the ending position, and an optional tag  $y$  associate with the segment.

Defining  $V(i, y)$  which represents the best sub-segmentation that ends with  $x_i$  (not included) and  $V(i, y)$  can be calculated as:

$$V(i, y) = \begin{cases} \max_{y', d=1 \dots L} V(i-d, y') + score(i-d, i, y), & \text{if } i > 0 \\ 0, & \text{if } i = 0 \\ -\infty, & \text{if } i < 0 \end{cases}$$

**for**  $i \leftarrow 1 \dots n$

**for**  $y \in \mathcal{Y}$ :

**for**  $d \leftarrow 1 \dots L$

            if  $i - d = 0$ :

$V(i, y) \leftarrow score(i-d, i, y)$

            else:

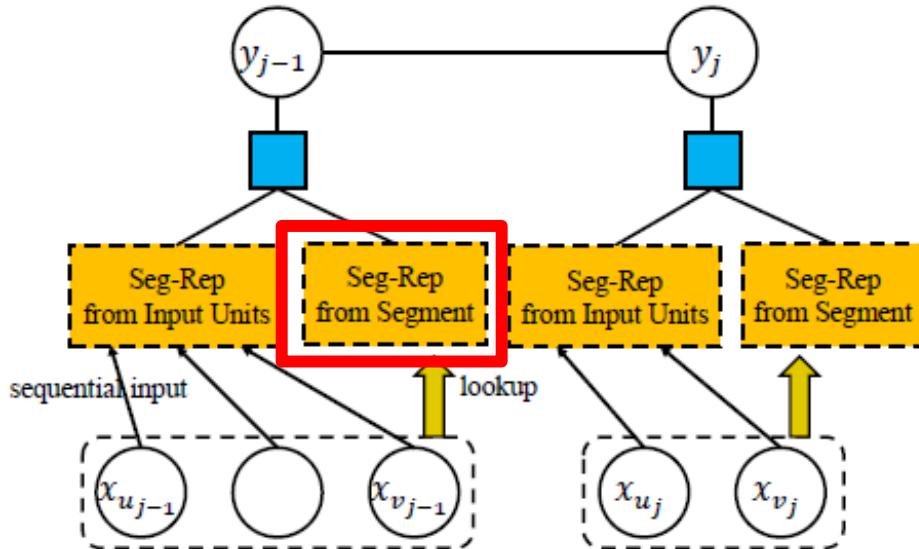
$best_{i-d} \leftarrow \max_{y'} V(i-d, y')$

$V(i, y) \leftarrow \max(V(i, y), best_{i-d} + score(i-d, i, y))$

# Results

		NER CoNLL03		CTB6				CWS PKU				MSR	
		dev	test	dev	test	dev	test	dev	test	dev	test	spd	
<i>baseline</i>	NN-LABELER	93.03	88.62	93.70	93.06	93.57	92.99	93.22	93.79	<b>3.30</b>			
	NN-CRF	<b>93.06</b>	<b>89.08</b>	94.33	93.65	94.09	93.28	93.81	94.17	2.72			
	SPARSE-CRF	88.87	83.43	<b>95.68</b>	<b>95.08</b>	<b>95.85</b>	<b>95.06</b>	<b>96.09</b>	<b>96.54</b>				
<i>neural semi-CRF</i>	SRNN	92.97	88.63	94.56	94.06	94.86	93.91	94.38	95.21	0.62			
	SCONCAT	92.96	89.07	94.34	93.96	94.41	93.57	94.05	94.53	1.08			
	SCNN	91.53	87.68	87.82	87.51	79.64	80.75	85.04	85.79	1.46			

# Segment-level Representation

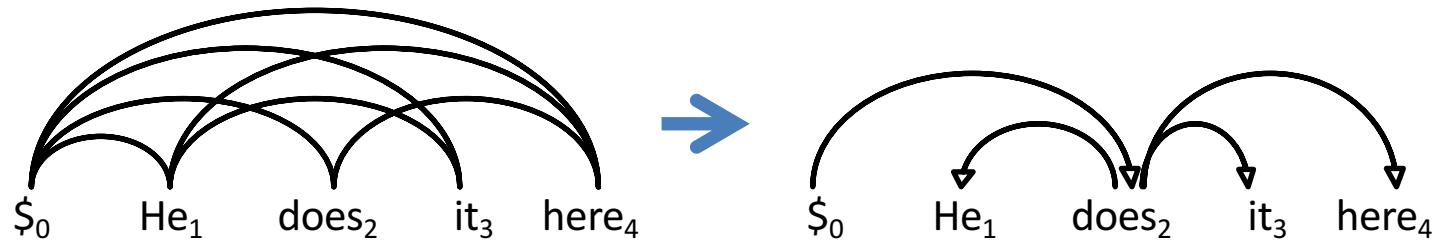


<i>model</i>	CoNLL03	CTB6	PKU	MSR
NN-LABELER	88.62	93.06	92.99	93.79
NN-CRF	89.08	93.65	93.28	94.17
SPARSE-CRF	83.43	95.08	95.06	96.54
SRNN	88.63	94.06	93.91	95.21
+SEMB-HETERO	89.59 +0.96	<b>95.48</b> +1.42	95.60 +1.69	97.39 +2.18
SCONCATÉ	89.07	93.96	93.57	94.53
+SEMB-HETERO	<b>89.77</b> +0.70	<b>95.42</b> +1.43	<b>95.67</b> +2.10	<b>97.58</b> +3.05

# Part 4.2: Dynamic Programming Decoding for Parsing

# Graph-based Dependency Parsing

- Find the highest scoring tree from a complete graph
- Dynamic Programming Decoding
  - E.g. Eisner Algorithm



$$Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y)$$

# How to Score an Arc?

$$score(6,1) = \mathbf{w} \cdot \mathbf{f}(6,1)$$



*	As	McGwire	neared	,	fans	went	wild		
	[went]		[VBD]		[As]		[ADP]		[went]
	[VERB]		[As]		[IN]		[went, VBD]		[As, ADP]
	[went, As]		[VBD, ADP]		[went, VERB]		[As, IN]		[went, As]
	[VERB, IN]		[VBD, As, ADP]		[went, As, ADP]		[went, VBD, ADP]		[went, VBD, As]
	[ADJ, *, ADP]		[VBD, *, ADP]		[VBD, ADJ, ADP]		[VBD, ADJ, *]		[NNS, *, ADP]
	[NNS, VBD, ADP]		[NNS, VBD, *]		[ADJ, ADP, NNP]		[VBD, ADP, NNP]		[VBD, ADJ, NNP]
	[NNS, ADP, NNP]		[NNS, VBD, NNP]		[went, left, 5]		[VBD, left, 5]		[As, left, 5]
	[ADP, left, 5]		[VERB, As, IN]		[went, As, IN]		[went, VERB, IN]		[went, VERB, As]
	[JJ, *, IN]		[VERB, *, IN]		[VERB, JJ, IN]		[VERB, JJ, *]		[NOUN, *, IN]
	[NOUN, VERB, IN]		[NOUN, VERB, *]		[JJ, IN, NOUN]		[VERB, IN, NOUN]		[VERB, JJ, NOUN]
	[NOUN, IN, NOUN]		[NOUN, VERB, NOUN]		[went, left, 5]		[VERB, left, 5]		[As, left, 5]
	[IN, left, 5]		[went, VBD, As, ADP]		[VBD, ADJ, *, ADP]		[NNS, VBD, *, ADP]		[VBD, ADJ, ADP, NNP]
	[NNS, VBD, ADP, NNP]		[went, VBD, left, 5]		[As, ADP, left, 5]		[went, As, left, 5]		[VBD, ADP, left, 5]
	[went, VERB, As, IN]		[VERB, JJ, *, IN]		[NOUN, VERB, *, IN]		[VERB, JJ, IN, NOUN]		[NOUN, VERB, IN, NOUN]
	[went, VERB, left, 5]		[As, IN, left, 5]		[went, As, left, 5]		[VERB, IN, left, 5]		[VBD, As, ADP, left, 5]
	[went, As, ADP, left, 5]		[went, VBD, ADP, left, 5]		[went, VBD, As, left, 5]		[ADJ, *, ADP, left, 5]		[VBD, *, ADP, left, 5]
	[VBD, ADJ, ADP, left, 5]		[VBD, ADJ, *, left, 5]		[NNS, *, ADP, left, 5]		[NNS, VBD, ADP, left, 5]		[NNS, VBD, *, left, 5]
	[ADJ, ADP, NNP, left, 5]		[VBD, ADP, NNP, left, 5]		[VBD, ADJ, NNP, left, 5]		[NNS, ADP, NNP, left, 5]		[NNS, VBD, NNP, left, 5]
	[VERB, As, IN, left, 5]		[went, As, IN, left, 5]		[went, VERB, IN, left, 5]		[went, VERB, As, left, 5]		[JJ, *, IN, left, 5]
	2016-10-14 [VERB, *, IN, left, 5]		[VERB, JJ, IN, left, 5]		[VERB, QL 20165 Tutorial]		[NOUN, *, IN, left, 5]		[NOUN, VERB, IN, left, 5]

# NN for Graph-based Parsing

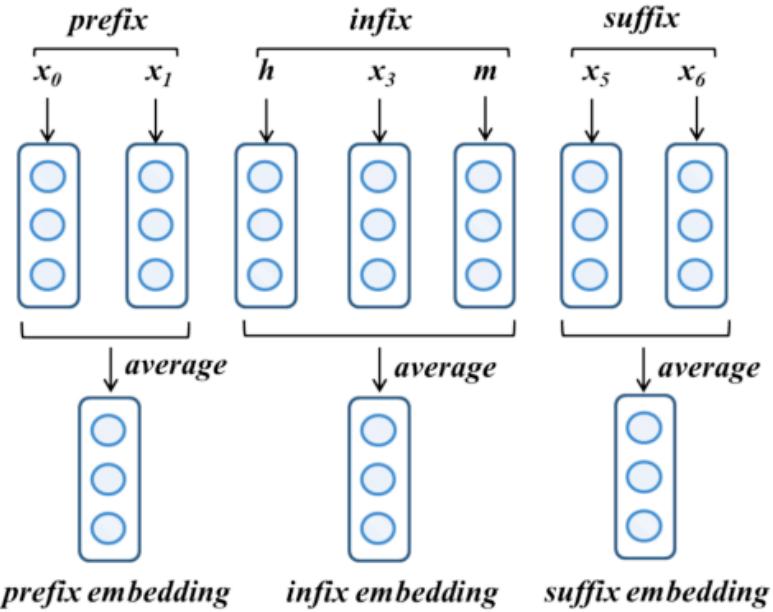
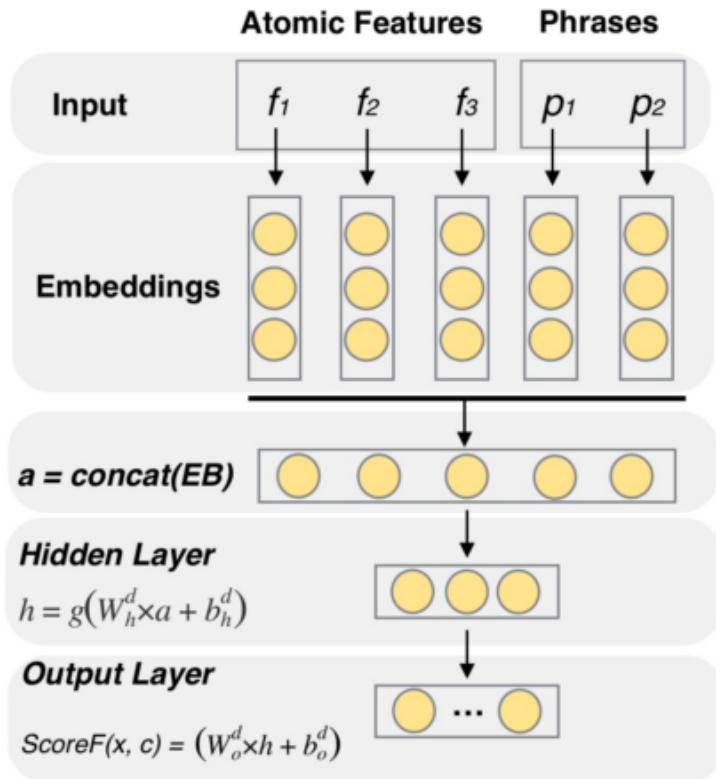


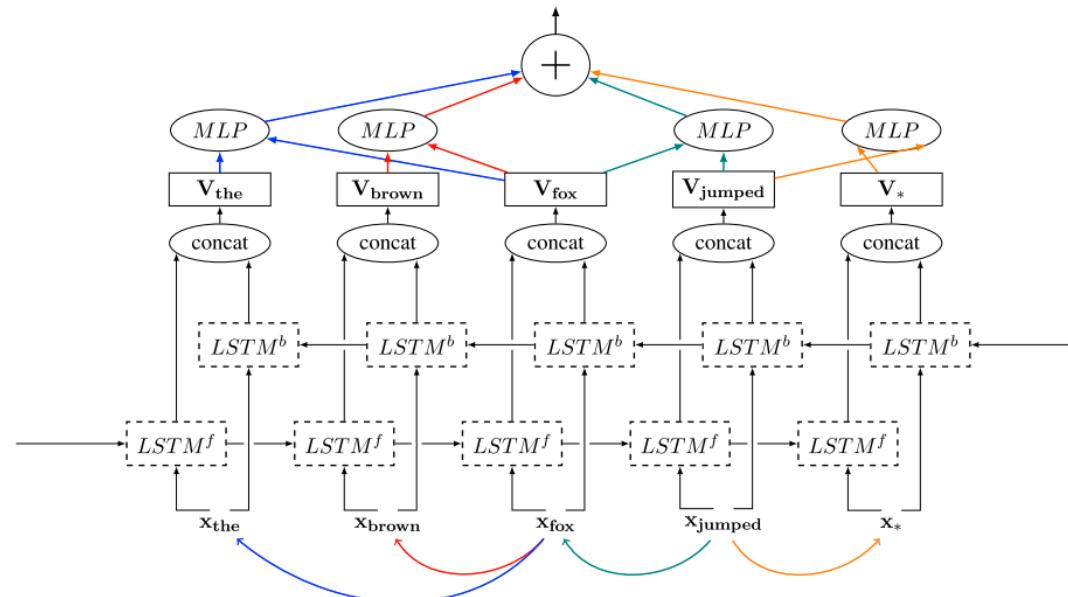
Figure 3: Illustration for phrase embeddings.  $h, m$  and  $x_0$  to  $x_6$  are words in the sentence.

# Results

	Models	Dev		Test		Speed (sent/s)
		UAS	LAS	UAS	LAS	
First-order	MSTParser-1-order	92.01	90.77	91.60	90.39	20
	<b>1-order-atomic-rand</b>	92.00	90.71	91.62	90.41	<b>55</b>
	<b>1-order-atomic</b>	92.19	90.94	92.14	90.92	<b>55</b>
	<b>1-order-phrase-rand</b>	92.47	91.19	92.25	91.05	26
	<b>1-order-phrase</b>	<b>92.82</b>	<b>91.48</b>	<b>92.59</b>	<b>91.37</b>	26
Second-order	MSTParser-2-order	92.70	91.48	92.30	91.06	14
	<b>2-order-phrase-rand</b>	93.39	92.10	92.99	91.79	10
	<b>2-order-phrase</b>	<b>93.57</b>	<b>92.29</b>	<b>93.29</b>	<b>92.13</b>	10
Third-order	(Koo and Collins, 2010)	93.49	N/A	93.04	N/A	N/A

# BI-LSTM for Graph-based Parsing-I

- Each dependency arc in a sentence is scored using MLP that is fed the BI-LSMT encoding of the words at the arc's end points

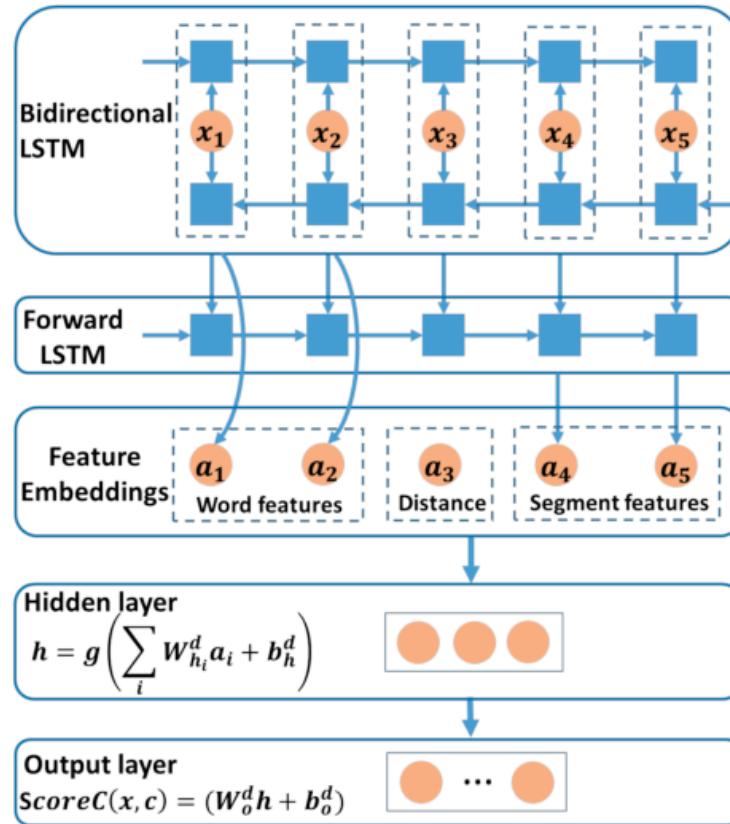


# Results

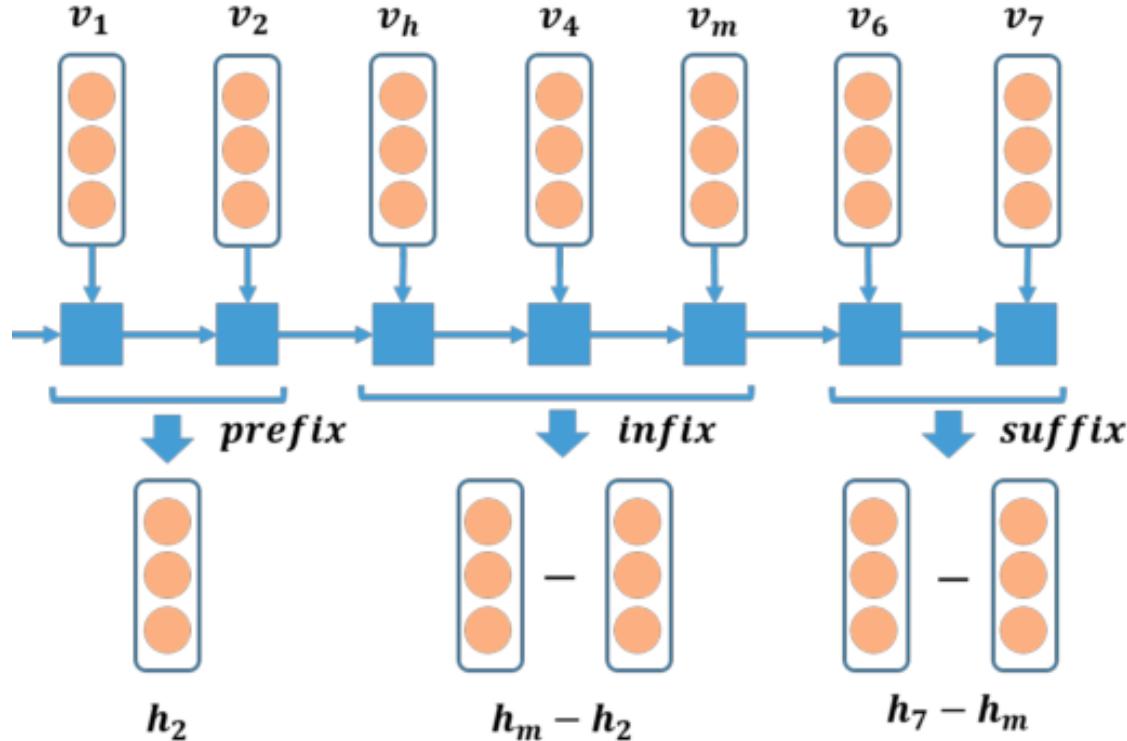
System	Method	Representation	Emb	PTB-YM		PTB-SD		CTB	
				UAS	LAS	UAS	LAS	UAS	LAS
This work	graph, 1st order	2 BiLSTM vectors	–	–	–	93.1	91.0	<b>86.6</b>	<b>85.1</b>
This work	transition (greedy, dyn-oracle)	4 BiLSTM vectors	–	–	–	93.1	91.0	86.2	85.0
This work	transition (greedy, dyn-oracle)	11 BiLSTM vectors	–	–	–	<b>93.2</b>	<b>91.2</b>	86.5	84.9
ZhangNivre11	transition (beam)	large feature set (sparse)	–	92.9	–	–	–	86.0	84.4
Martins13 (TurboParser)	graph, 3rd order+	large feature set (sparse)	–	92.8	93.1	–	–	–	–
Pei15	graph, 2nd order	large feature set (dense)	–	93.0	–	–	–	–	–
Dyer15	transition (greedy)	Stack-LSTM + composition	–	–	92.4	90.0	85.7	84.1	
Ballesteros16	transition (greedy, dyn-oracle)	Stack-LSTM + composition	–	–	92.7	90.6	86.1	84.5	
This work	graph, 1st order	2 BiLSTM vectors	YES	–	93.0	90.9	86.5	84.9	
This work	transition (greedy, dyn-oracle)	4 BiLSTM vectors	YES	–	93.6	91.5	87.4	85.9	
This work	transition (greedy, dyn-oracle)	11 BiLSTM vectors	YES	–	93.9	91.9	<b>87.6</b>	86.1	
Weiss15	transition (greedy)	large feature set (dense)	YES	–	93.2	91.2	–	–	
Weiss15	transition (beam)	large feature set (dense)	YES	–	<b>94.0</b>	<b>92.0</b>	–	–	
Pei15	graph, 2nd order	large feature set (dense)	YES	93.3	–	–	–	–	
Dyer15	transition (greedy)	Stack-LSTM + composition	YES	–	93.1	90.9	87.1	85.5	
Ballesteros16	transition (greedy, dyn-oracle)	Stack-LSTM + composition	YES	–	93.6	91.4	<b>87.6</b>	<b>86.2</b>	
LeZuidema14	reranking /blend	inside-outside recursive net	YES	93.1	93.8	91.5	–	–	
Zhu15	reranking /blend	recursive conv-net	YES	93.8	–	–	85.7	–	

# BI-LSTM for Graph-based Parsing-II

- Besides the word vectors, they used sentence segment (phrase) embeddings



# Learning Segment Embeddings



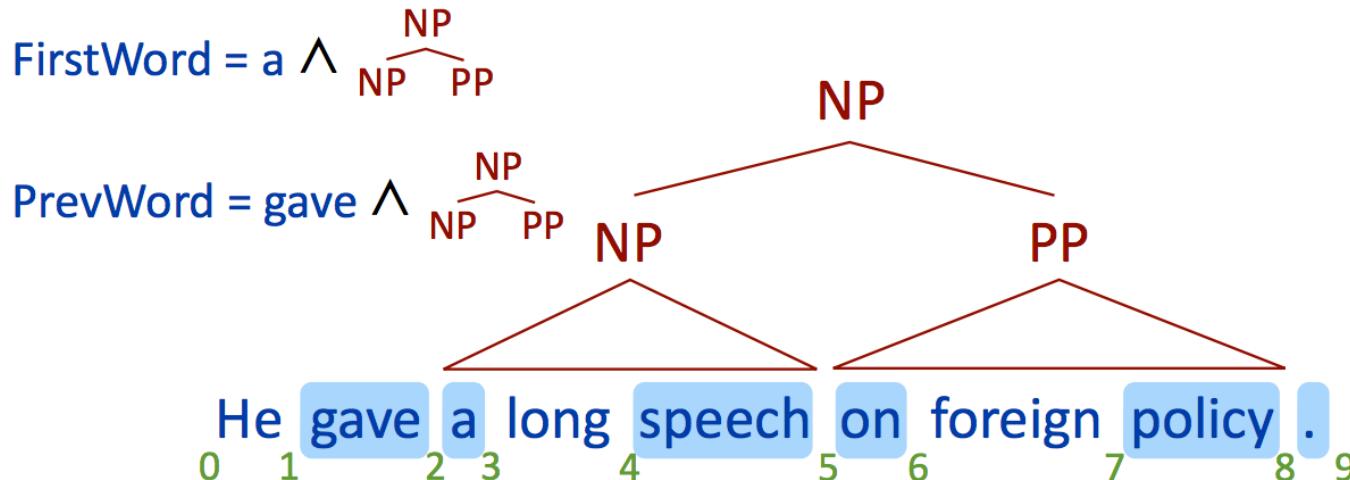
# Results

	Models	UAS	LAS	Speed(sent/s)
First-order	MSTParser	91.60	90.39	20
	1st-order atomic (Pei et al., 2015)	92.14	90.92	55
	1st-order phrase (Pei et al., 2015)	92.59	91.37	26
	<b>Our basic model</b>	93.09	92.03	<b>61</b>
	<b>Our basic model + segment</b>	<b>93.51</b>	<b>92.45</b>	26
Second-order	MSTParser	92.30	91.06	14
	2nd-order phrase (Pei et al., 2015)	93.29	92.13	10
Third-order	(Koo and Collins, 2010)	93.04	N/A	N/A
Fourth-order	(Ma and Zhao, 2012)	93.4	N/A	N/A
Unlimited-order	(Zhang and McDonald, 2012)	93.06	91.86	N/A
	(Zhang et al., 2013)	93.50	92.41	N/A
	<b>(Zhang and McDonald, 2014)</b>	<b>93.57</b>	<b>92.48</b>	N/A

# Neural CRF for Constituency Parsing

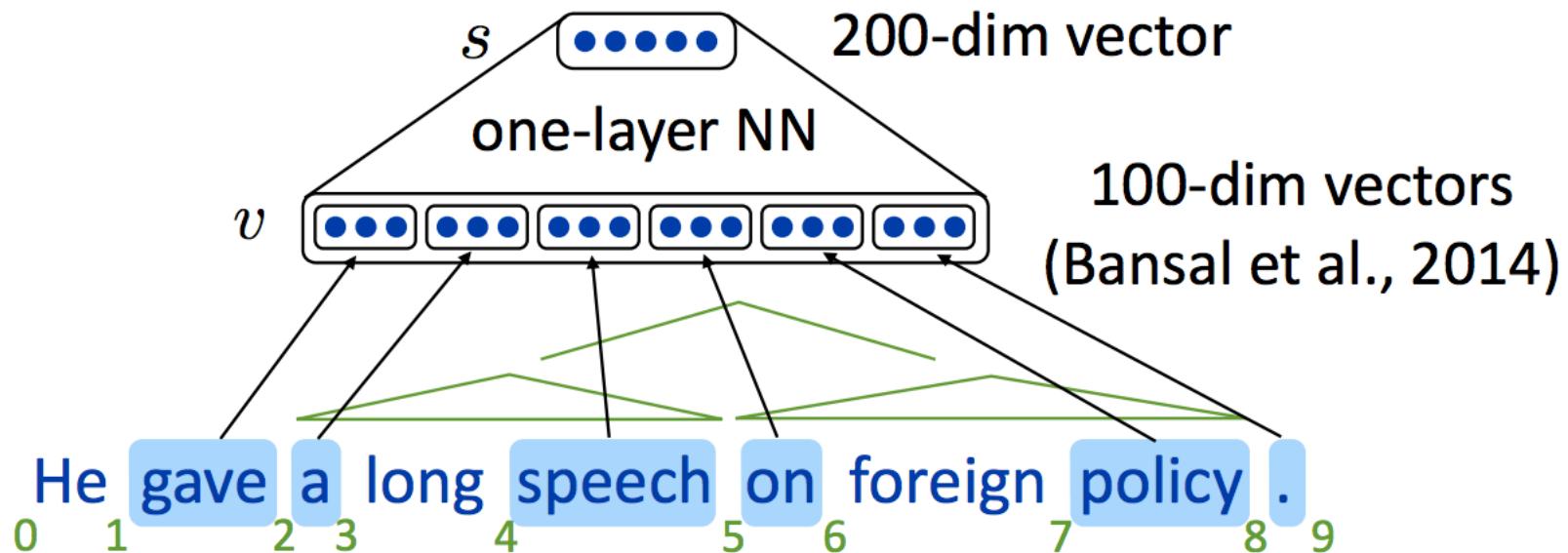
- CRF Parsing with CKY decoding

$$P(T|x) \propto \prod_{r \in T} \exp(\text{score}(r)) \quad \text{score}\left(\begin{array}{ccccc} & \text{NP} & & & \\ & \diagdown & \diagup & & \\ 2 & \text{NP} & 5 & \text{PP} & 8 \end{array}\right) = w^\top f\left(\begin{array}{ccccc} & \text{NP} & & & \\ & \diagdown & \diagup & & \\ 2 & \text{NP} & 5 & \text{PP} & 8 \end{array}\right)$$

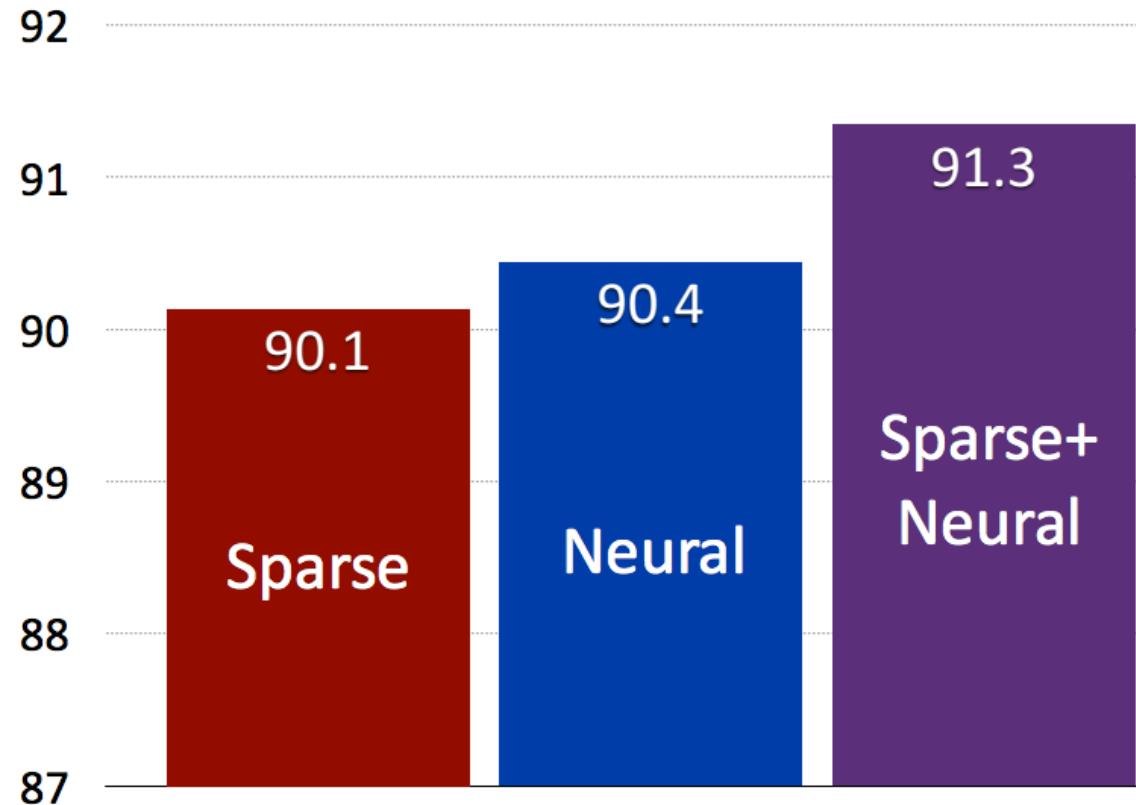


# Neural CRF for Constituency Parsing

- Neural CRF Parsing

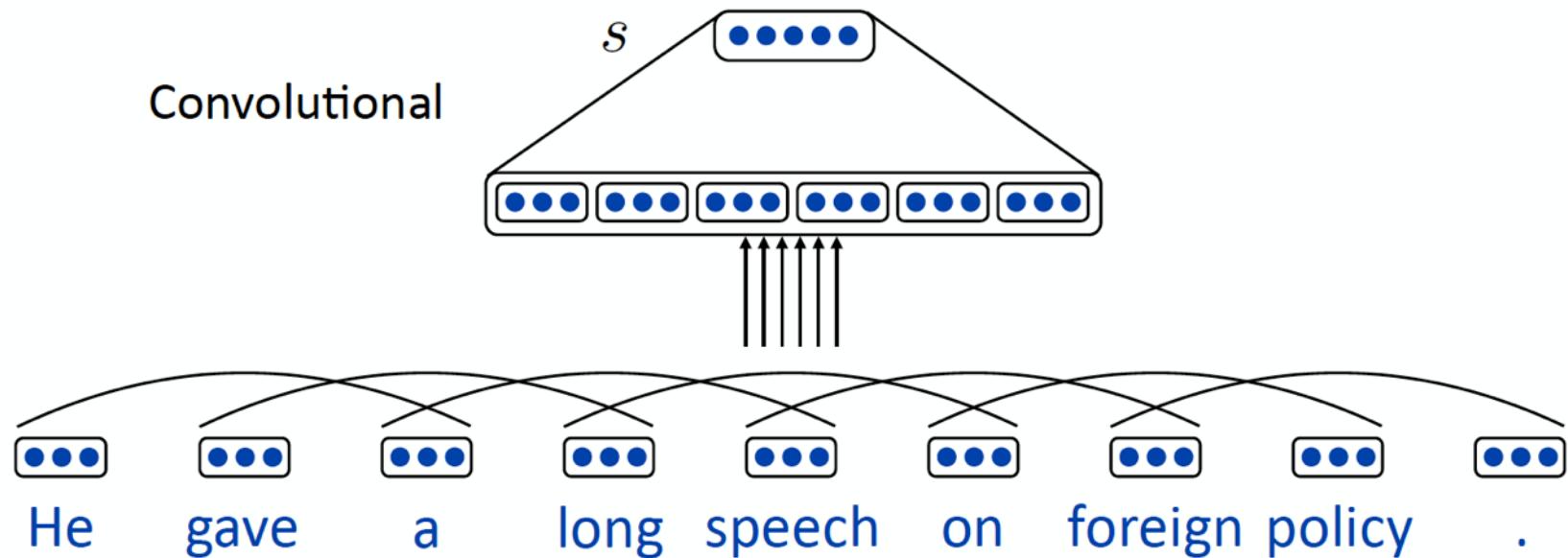


# Results



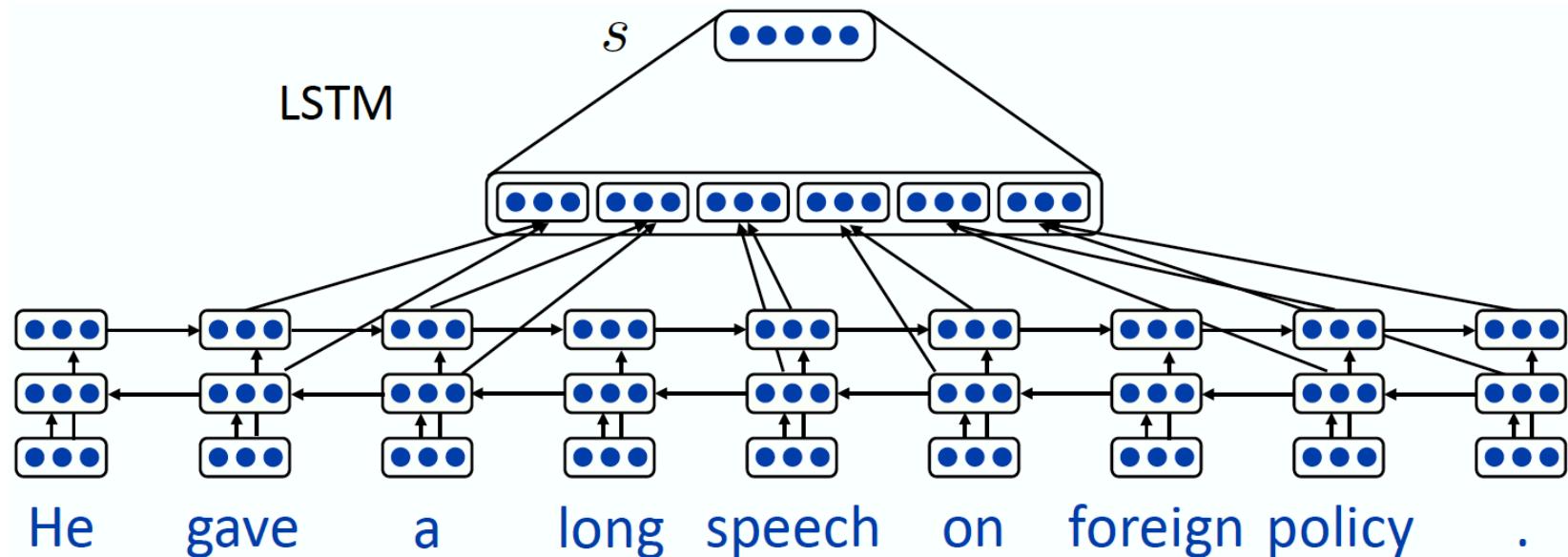
# Neural CRF for Constituency Parsing

- More neural networks



# Neural CRF for Constituency Parsing

- More neural networks



# Conclusion

- Neural nets can provide continuous features in discrete structured models
- Inference and learning are almost unchanged from the purely discrete model

# Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che (HIT)

Yue Zhang (SUTD)

# Part 5: Beam-search Decoding

# A transition system

- Automata
  - State
    - Start state —— an empty structure
    - End state —— the output structure
    - Intermediate states —— partially constructed structures
  - Actions
    - Change one state to another

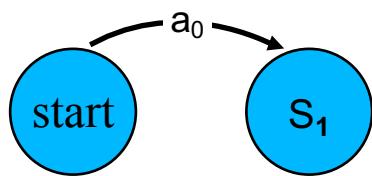
# A transition system

- Automata



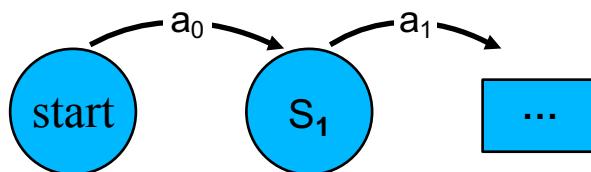
# A transition system

- Automata



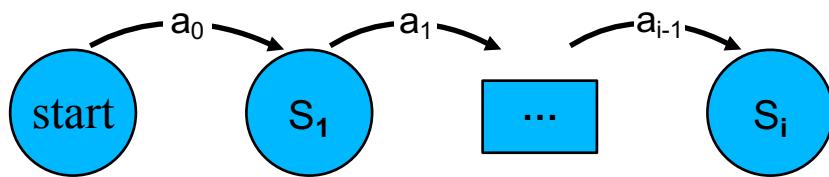
# A transition system

- Automata



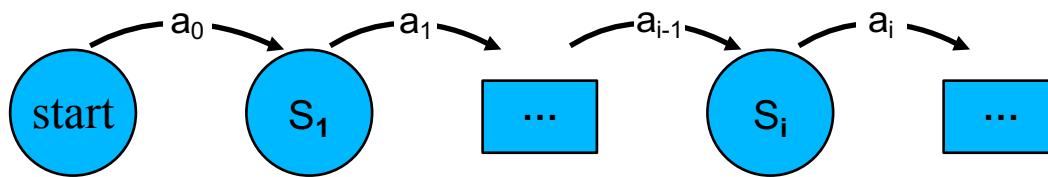
# A transition system

- Automata



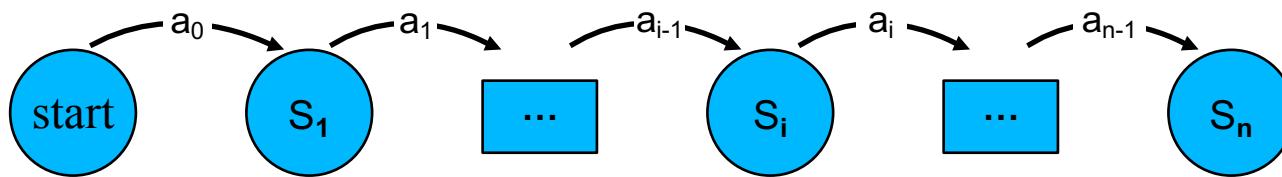
# A transition system

- Automata



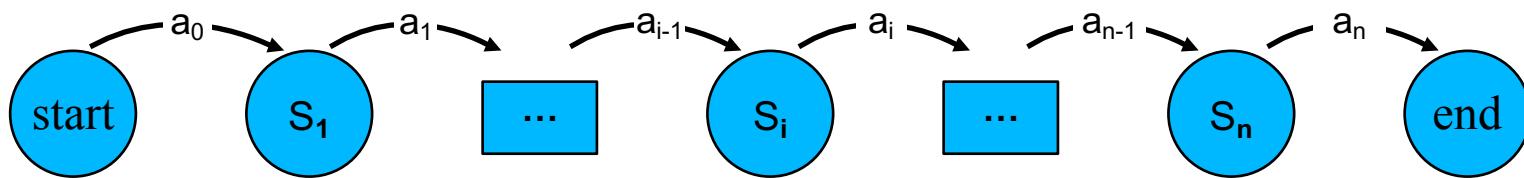
# A transition system

- Automata



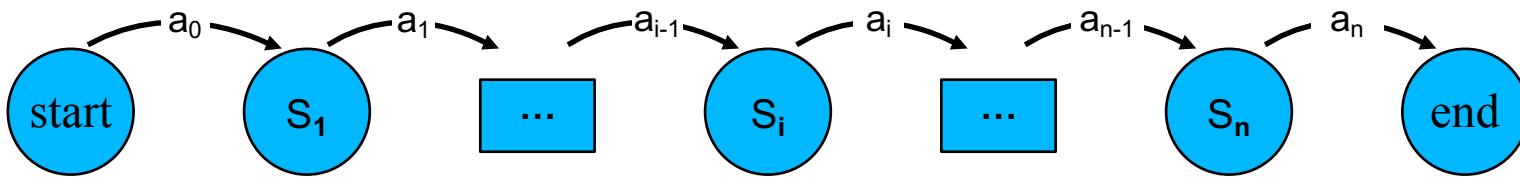
# A transition system

- Automata



# A transition system

- State
  - Corresponds to partial results during decoding
    - start state, end state,  $S_i$



- Actions
  - The operations that can be applied for state transition
  - Construct output incrementally
    - $a_i$

# A transition-based POS-tagging example

- POS tagging

I like reading books → I/PRON like/VERB reading/VERB books/NOUN

- Transition system

- State

- Partially labeled word-POS pairs
    - Unprocessed words

- Actions

- TAG(t)  $w_1/t_1 \cdots w_i/t_i \rightarrow w_1/t_1 \cdots w_i/t_i w_{i+1}/t$

# A transition-based POS-tagging example

- Start State



I like reading books

# A transition-based POS-tagging example

- TAG(PRON)

I/PRON

like reading books

# A transition-based POS-tagging example

- TAG(VERB)

I/PRON like/VERB

reading books

# A transition-based POS-tagging example

- TAG(VERB)

I/PRON like/VERB reading/VERB

books

# A transition-based POS-tagging example

- TAG (NOUN)

I/PRON like/VERB reading/VERB books/NOUN

# A transition-based POS-tagging example

- End State

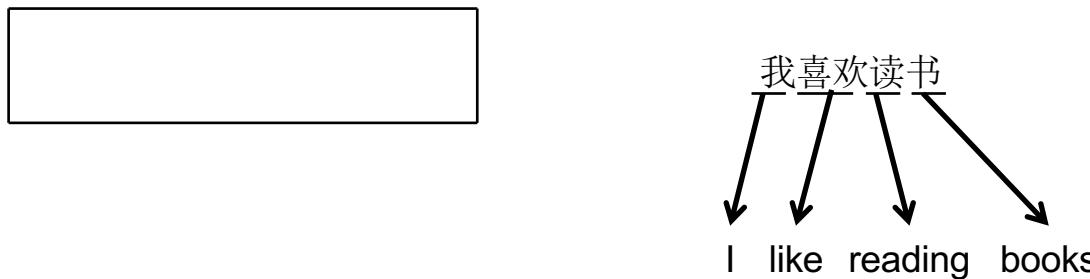
I/PRON like/VERB reading/VERB books/NOUN

# Word segmentation

- State
  - Partially segmented results
  - Unprocessed characters
- Two candidate actions
  - Separate       $\#\# \text{ } \#\# \rightarrow \#\# \text{ } \#\# \text{ } \#$
  - Append         $\#\# \text{ } \#\# \rightarrow \#\# \text{ } \#\# \text{ } \#$

# Word segmentation

- Initial State



# Word segmentation

- Separate

我

喜欢读书

# Word segmentation

- Separate

我 喜

欢读书

# Word segmentation

- Append

我 喜欢

读书

# Word segmentation

- Separate

我 喜欢 读

书

# Word segmentation

- Separate

我 喜欢 读 书

# Word segmentation

- End State

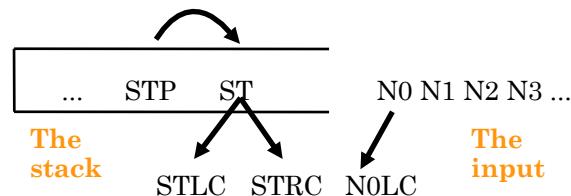
我 喜欢 读 书

# The arc-eager transition system

- State
  - A stack to hold partial structures
  - A queue of next incoming words
- Actions
  - SHIFT, REDUCE, ARC-LEFT, ARC-RIGHT

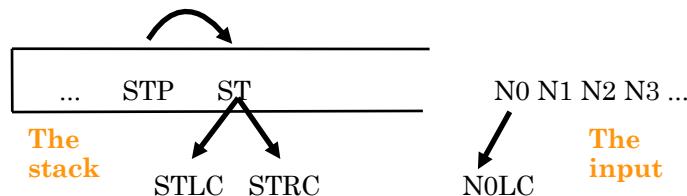
# The arc-eager transition system

- State



# The arc-eager transition system

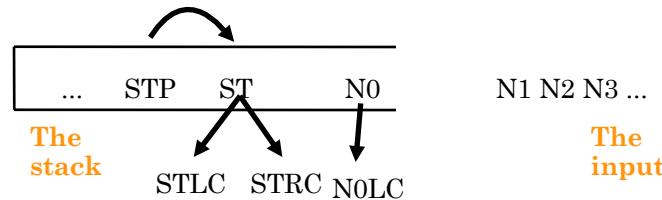
- Actions
  - Shift



# The arc-eager transition system

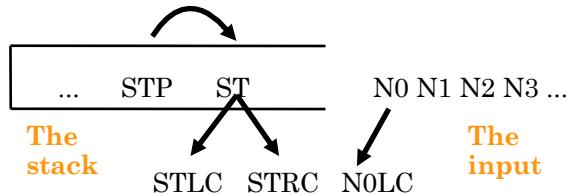
- Actions
  - Shift

➤ Pushes stack



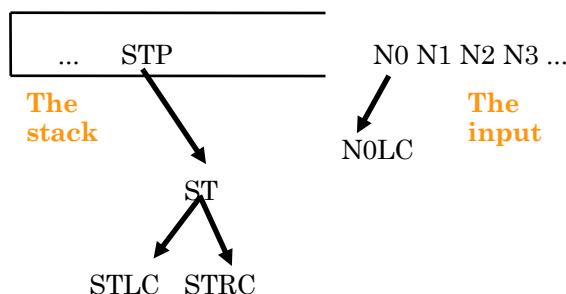
# The arc-eager transition system

- Actions
  - Reduce



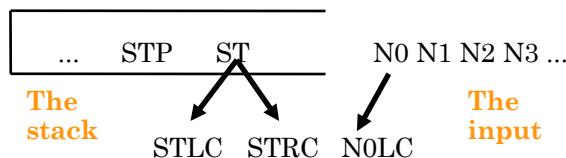
# The arc-eager transition system

- Actions
  - Reduce
    - Pops stack



# The arc-eager transition system

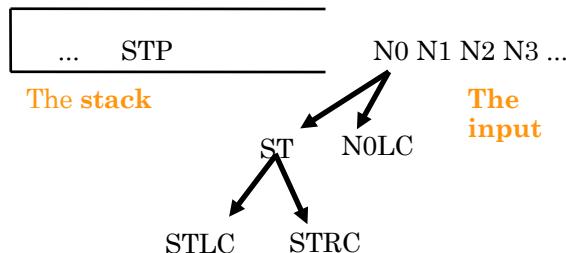
- Actions
  - Arc-Left



# The arc-eager transition system

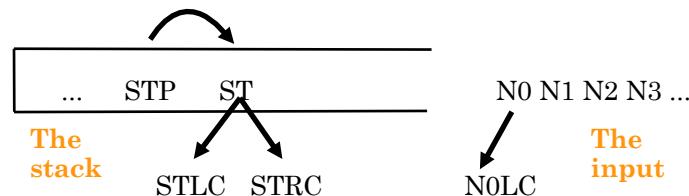
- Actions
  - Arc-Left

- Pops stack
- Adds link



# The arc-eager transition system

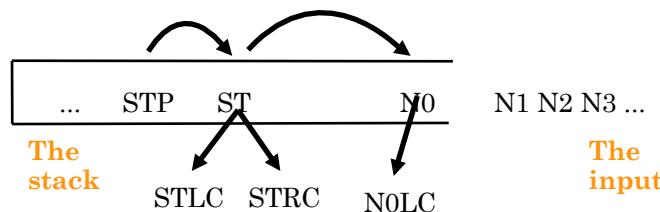
- Actions
  - Arc-right



# The arc-eager transition system

- Actions
  - Arc-right

- Pushes stack
- Adds link



# The arc-eager transition system

- An example

- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



He does it here

# The arc-eager transition system

- An example

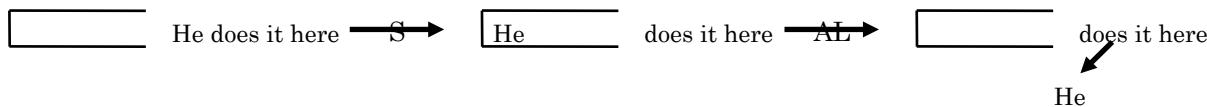
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight

\_\_\_\_\_ He does it here  $\xrightarrow{S}$  [He \_\_\_\_\_ does it here]

# The arc-eager transition system

- An example

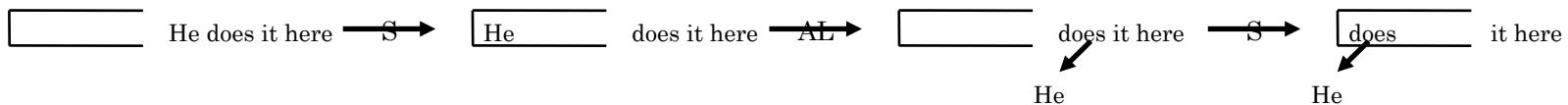
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



# The arc-eager transition system

- An example

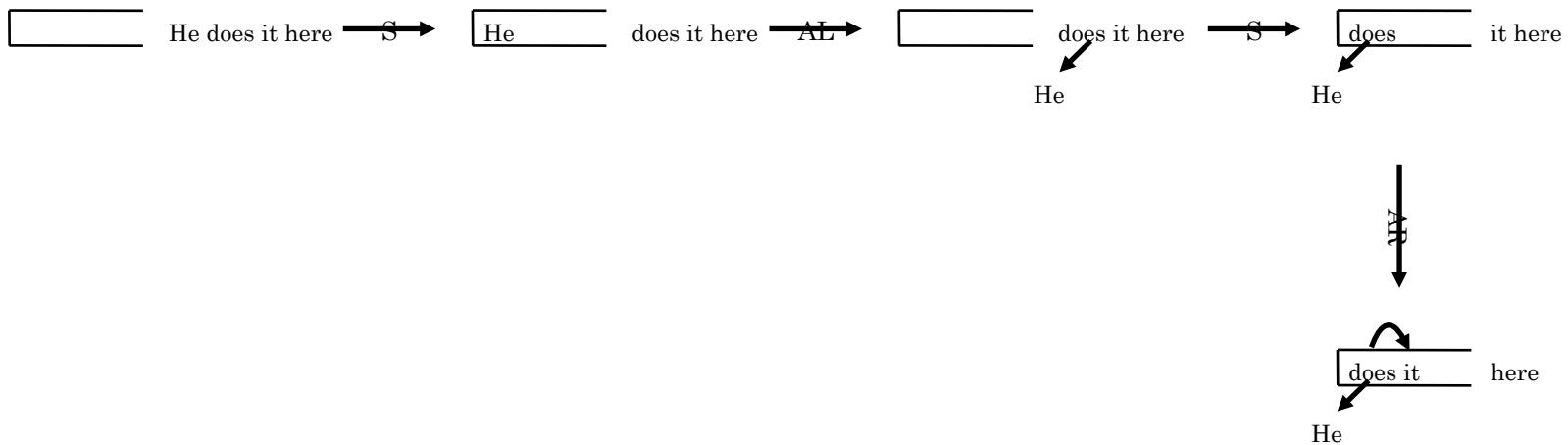
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



# The arc-eager transition system

- An example

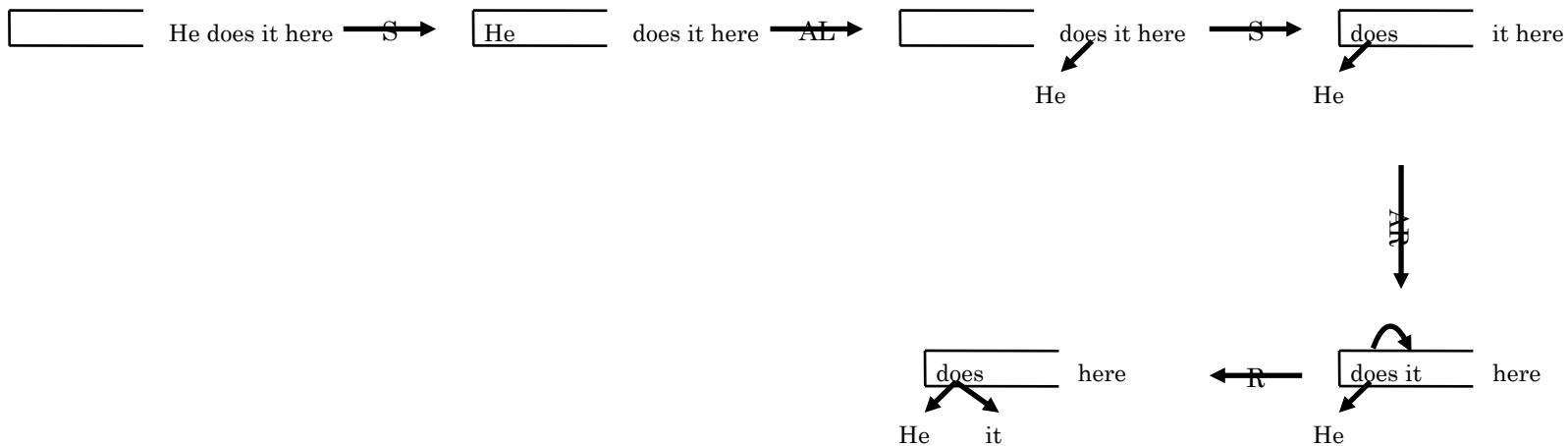
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



# The arc-eager transition system

- An example

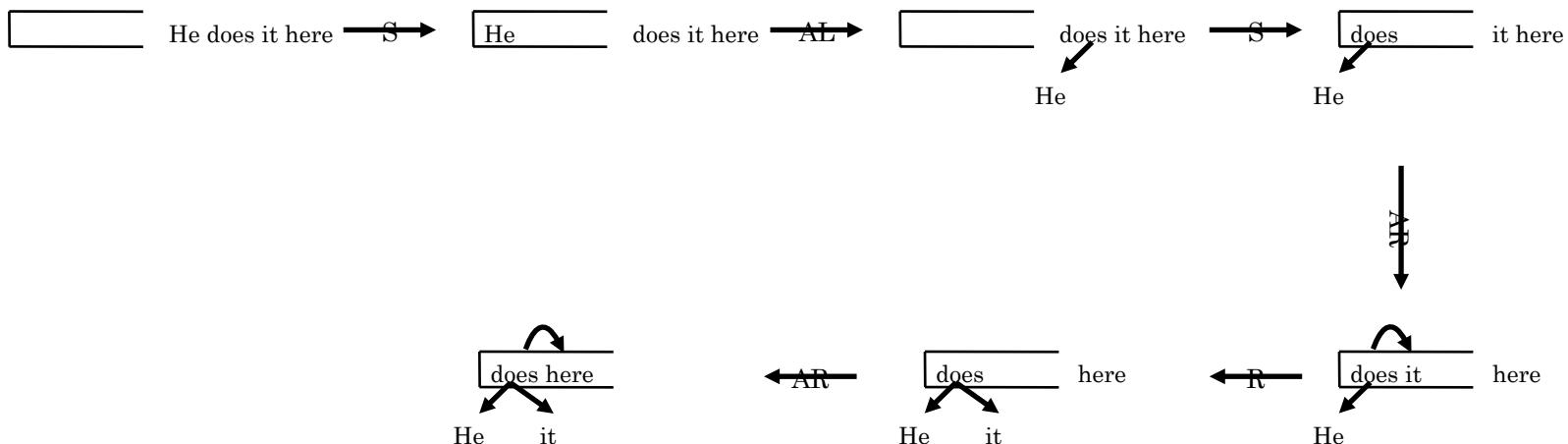
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



# The arc-eager transition system

- An example

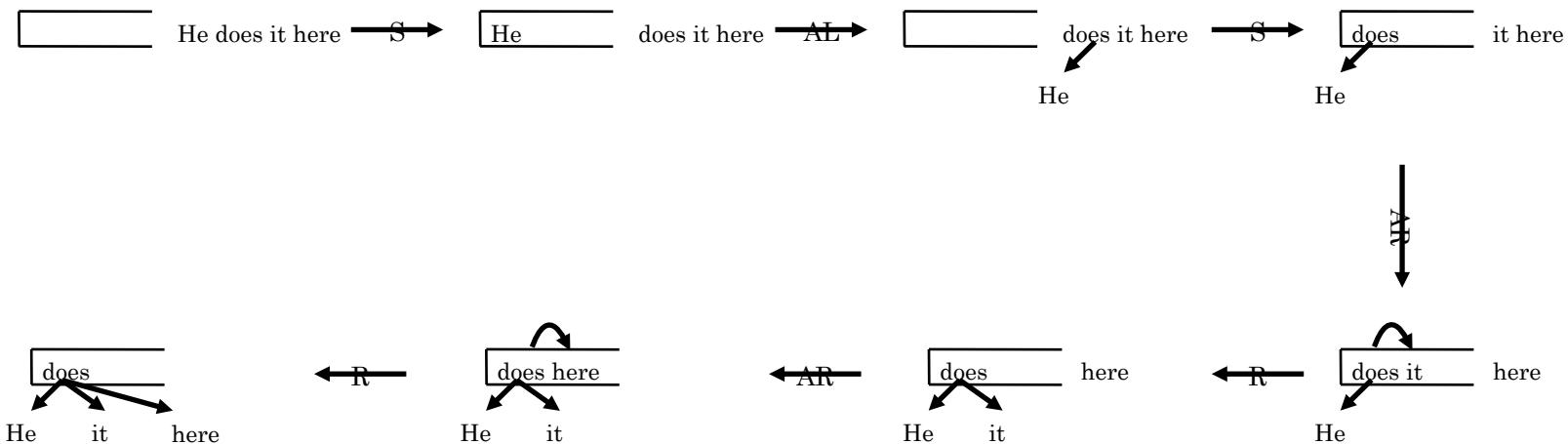
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



# The arc-eager transition system

- An example

- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight



# Other examples

- Language generation
- Translation
  - Word by word
  - Phrase by phrase
  - Syntax tree synthesis

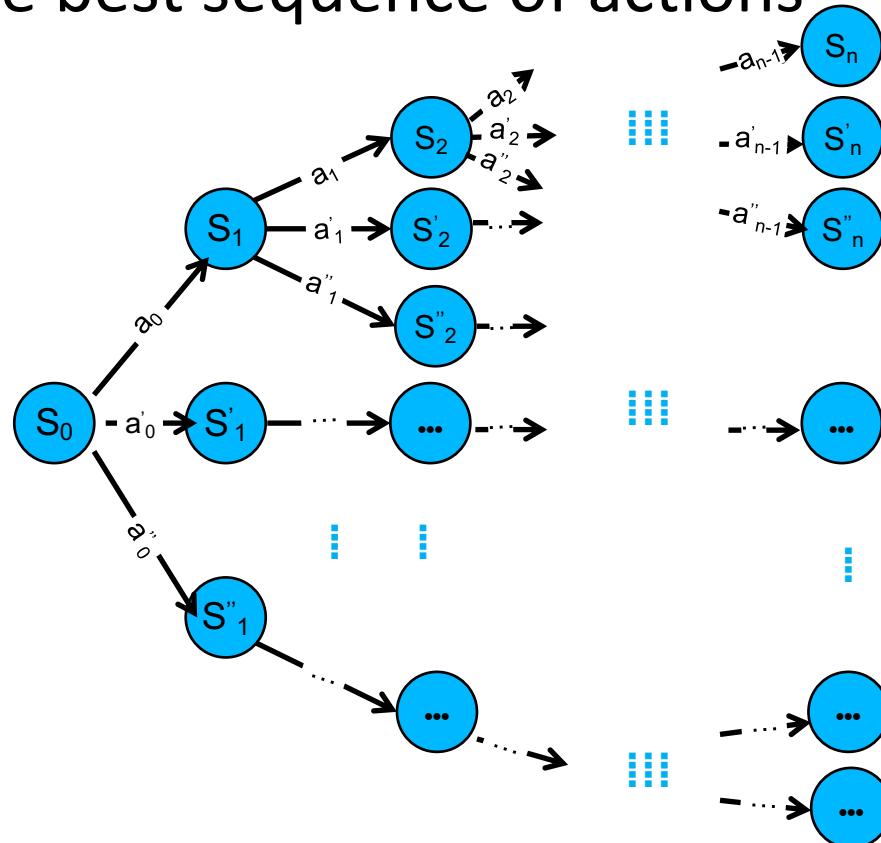
# Part 5.1: Beam-search Decoding

## ——learning to search

(Zhang and Clark,2011)

# Search

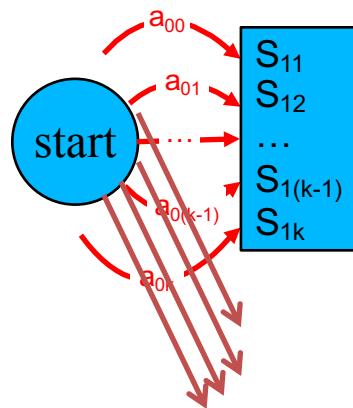
- Find the best sequence of actions



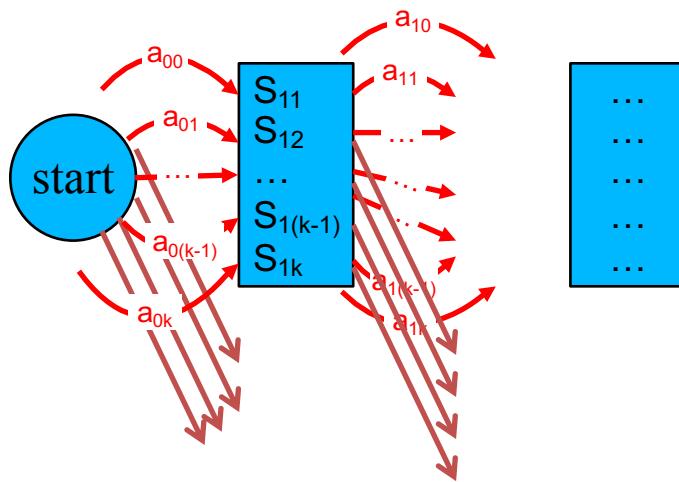
# Beam-search decoding



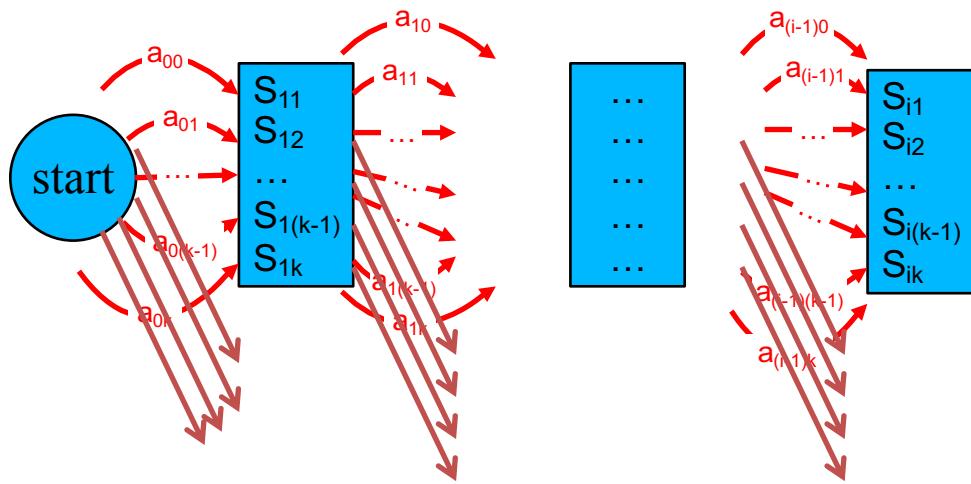
# Beam-search decoding



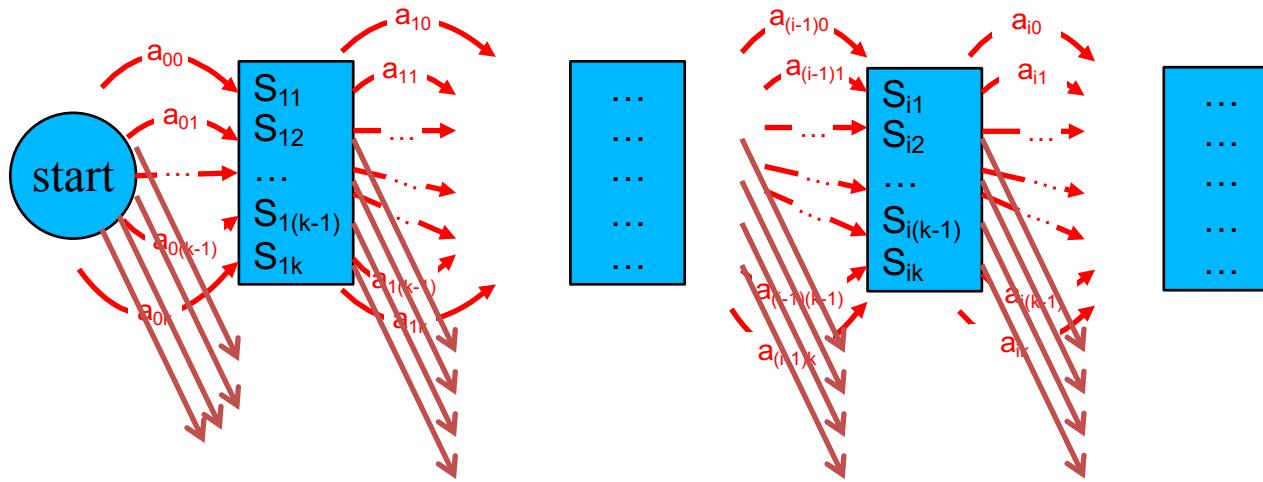
# Beam-search decoding



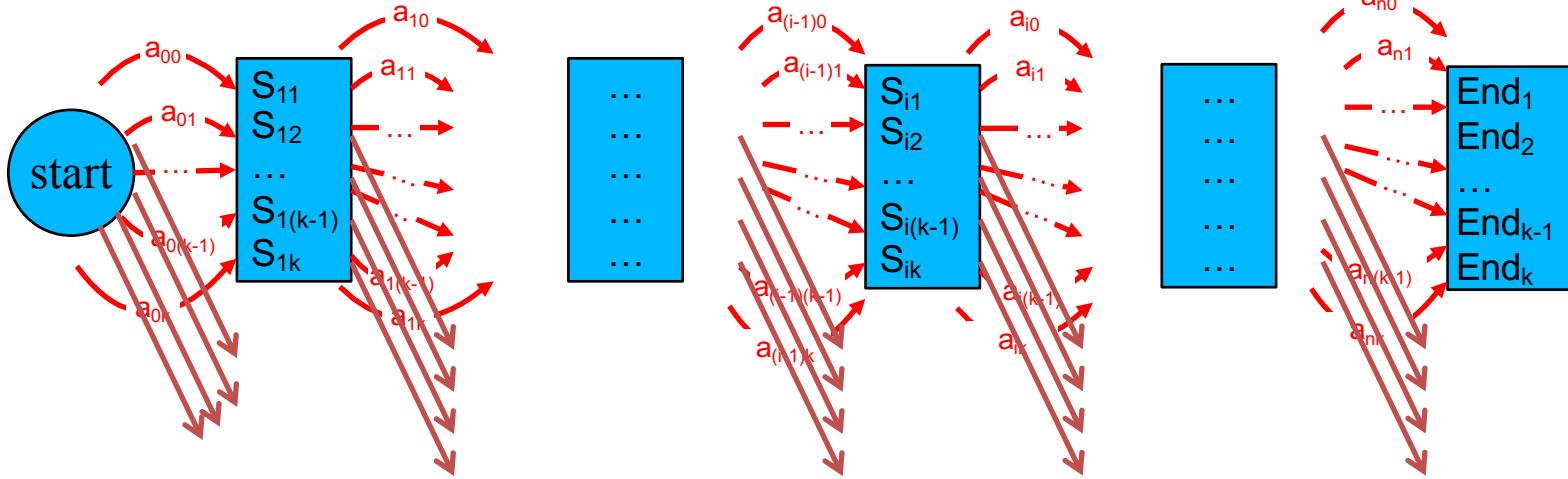
# Beam-search decoding



# Beam-search decoding



# Beam-search decoding

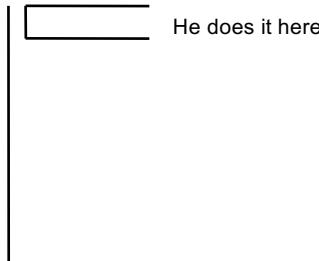


# Beam-search decoding

```
function BEAM-SEARCH(problem, agenda, candidates, B)  
  
    candidates  $\leftarrow \{\text{STARTITEM}(\textit{problem})\}$   
    agenda  $\leftarrow \text{CLEAR}(\textit{agenda})$   
    loop do  
        for each candidate in candidates  
            agenda  $\leftarrow \text{INSERT}(\text{EXPAND}(\textit{candidate}, \textit{problem}), \textit{agenda})$   
        best  $\leftarrow \text{TOP}(\textit{agenda})$   
        if GOALTEST(problem, best)  
            then return best  
        candidates  $\leftarrow \text{TOP-B}(\textit{agenda}, \textit{B})$   
        agenda  $\leftarrow \text{CLEAR}(\textit{agenda})$ 
```

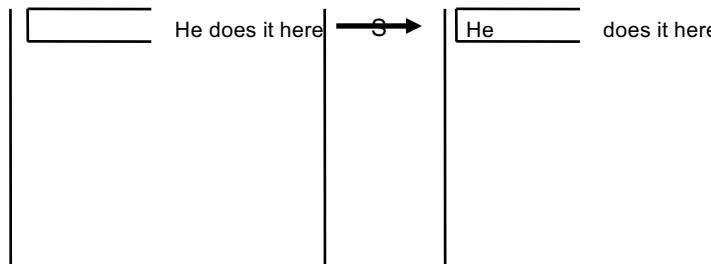
# Beam-search decoding

- Our parser
  - Decoding



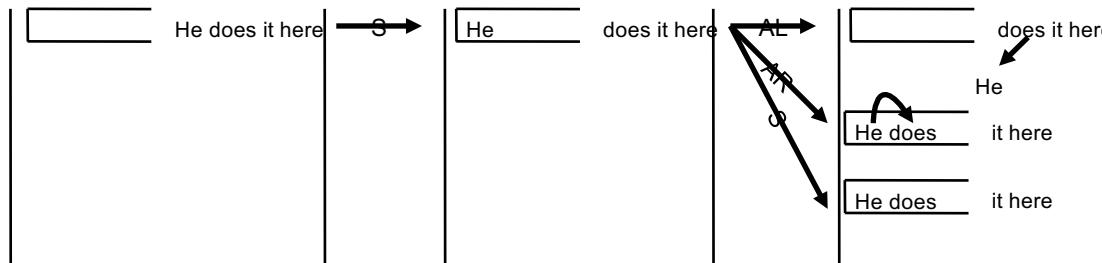
# Beam-search decoding

- Our parser
  - Decoding



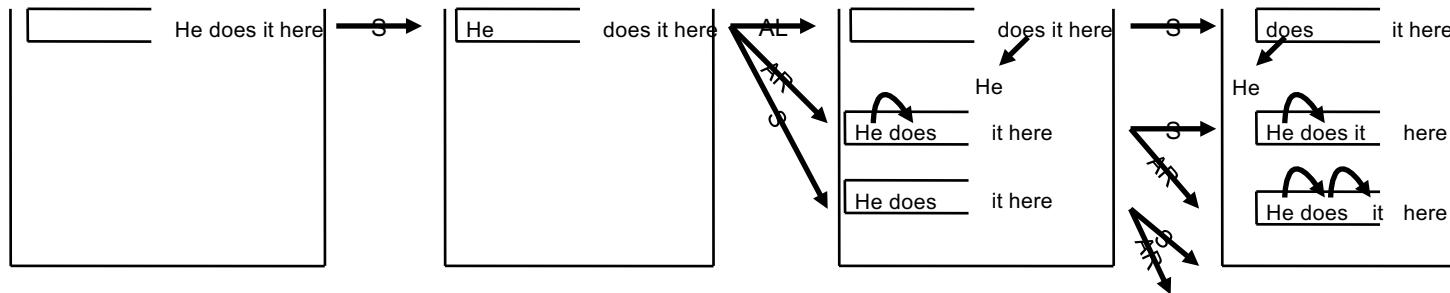
# Beam-search decoding

- Our parser
  - Decoding



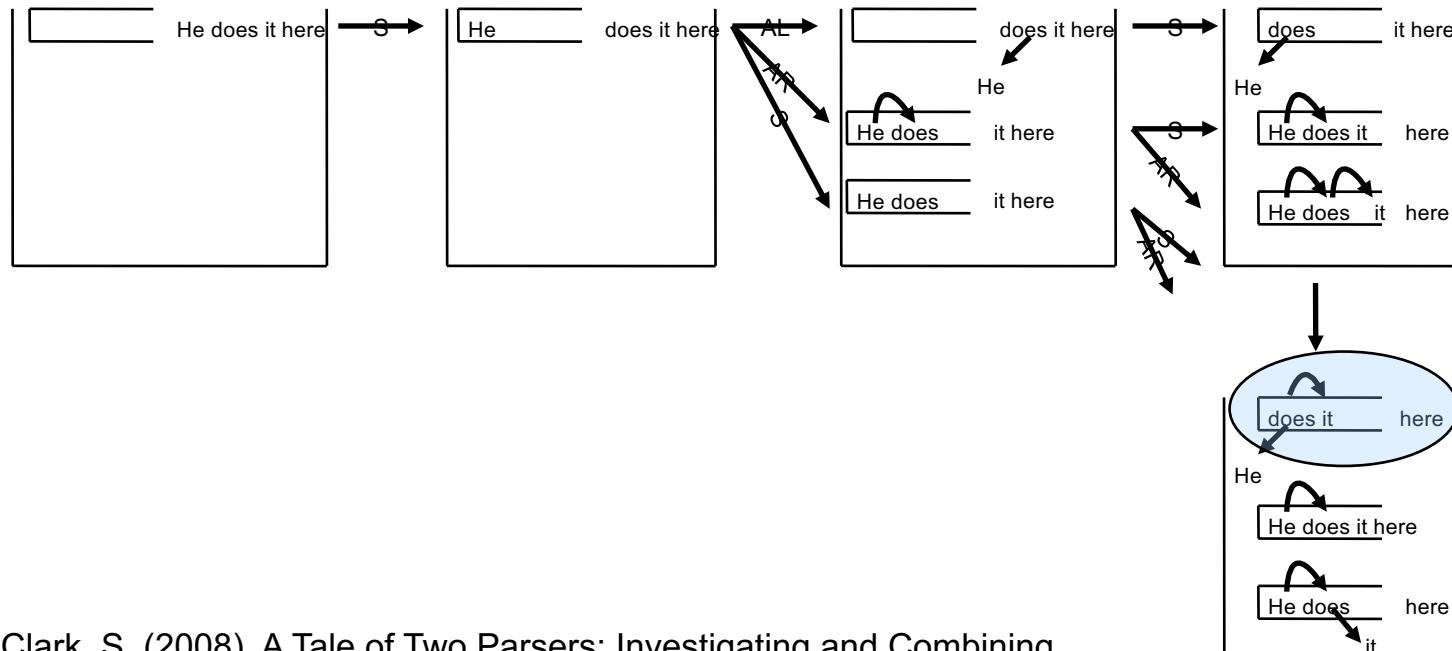
# Beam-search decoding

- Our parser
  - Decoding



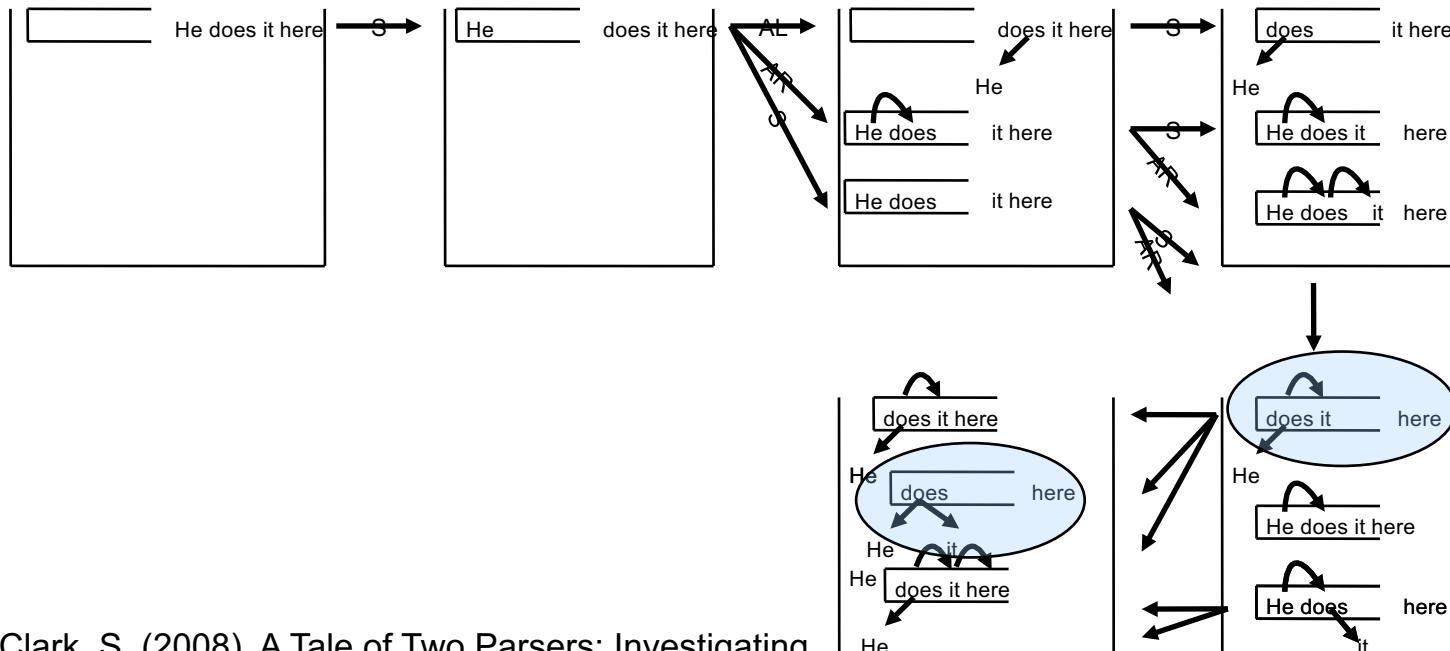
# Beam-search decoding

- Our parser
  - Decoding



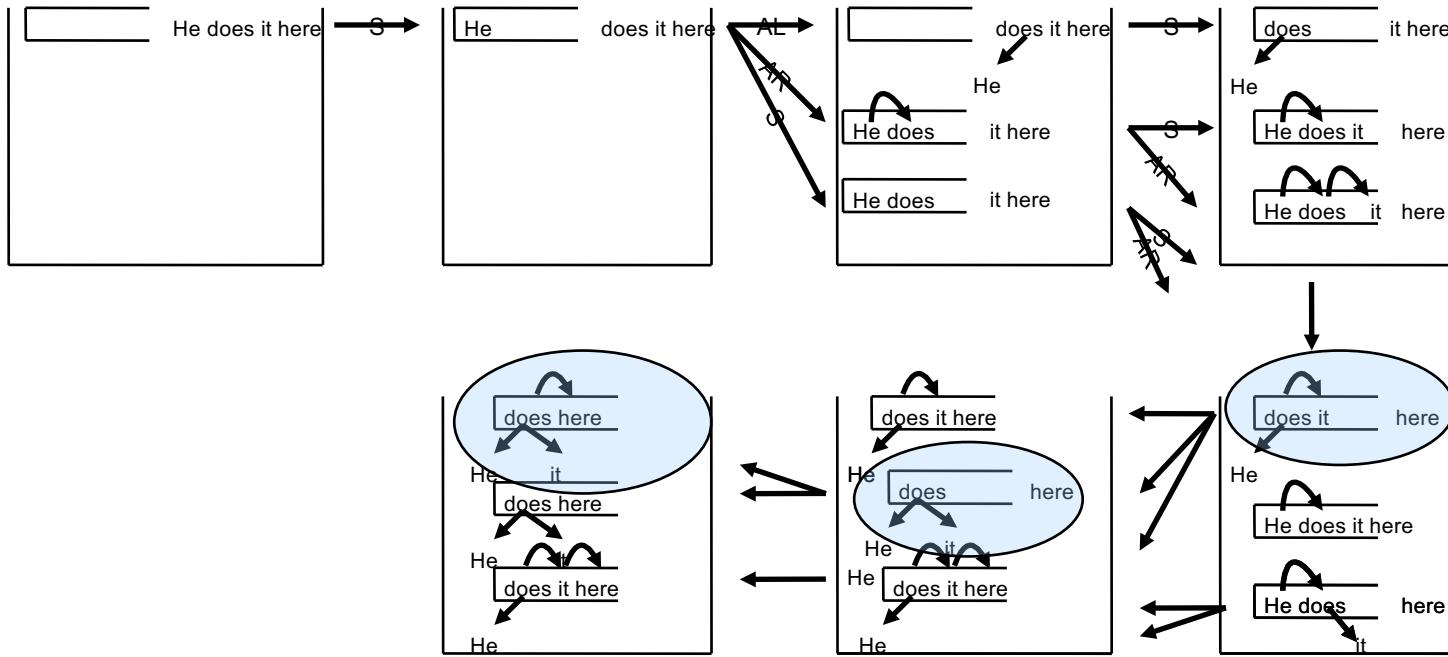
# Beam-search decoding

- Our parser
  - Decoding



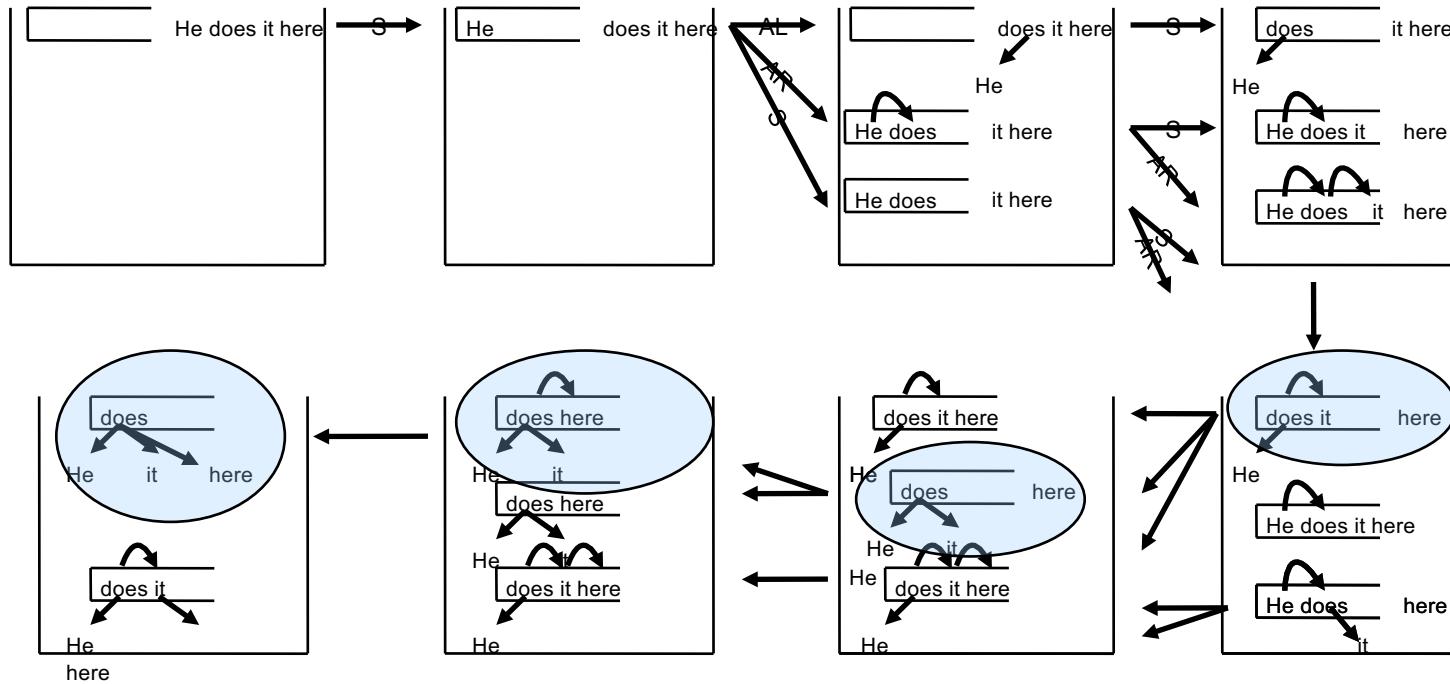
# Beam-search decoding

- Our parser
  - Decoding



# Beam-search decoding

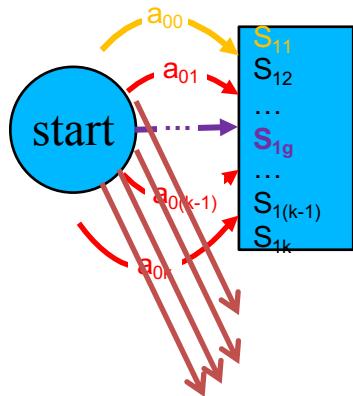
- Our parser
  - Decoding



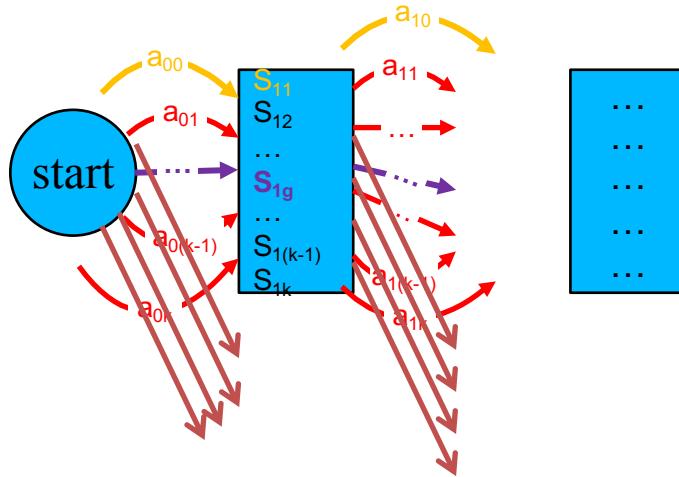
# Online learning



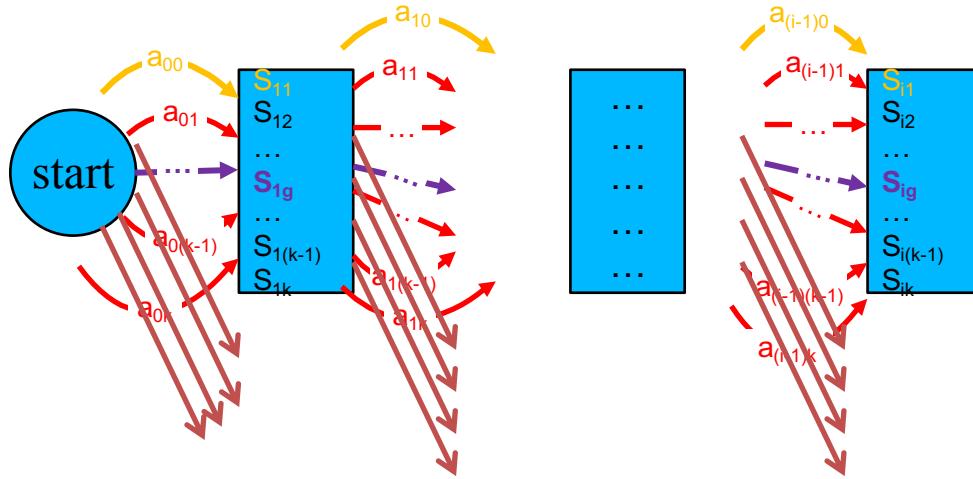
# Online learning



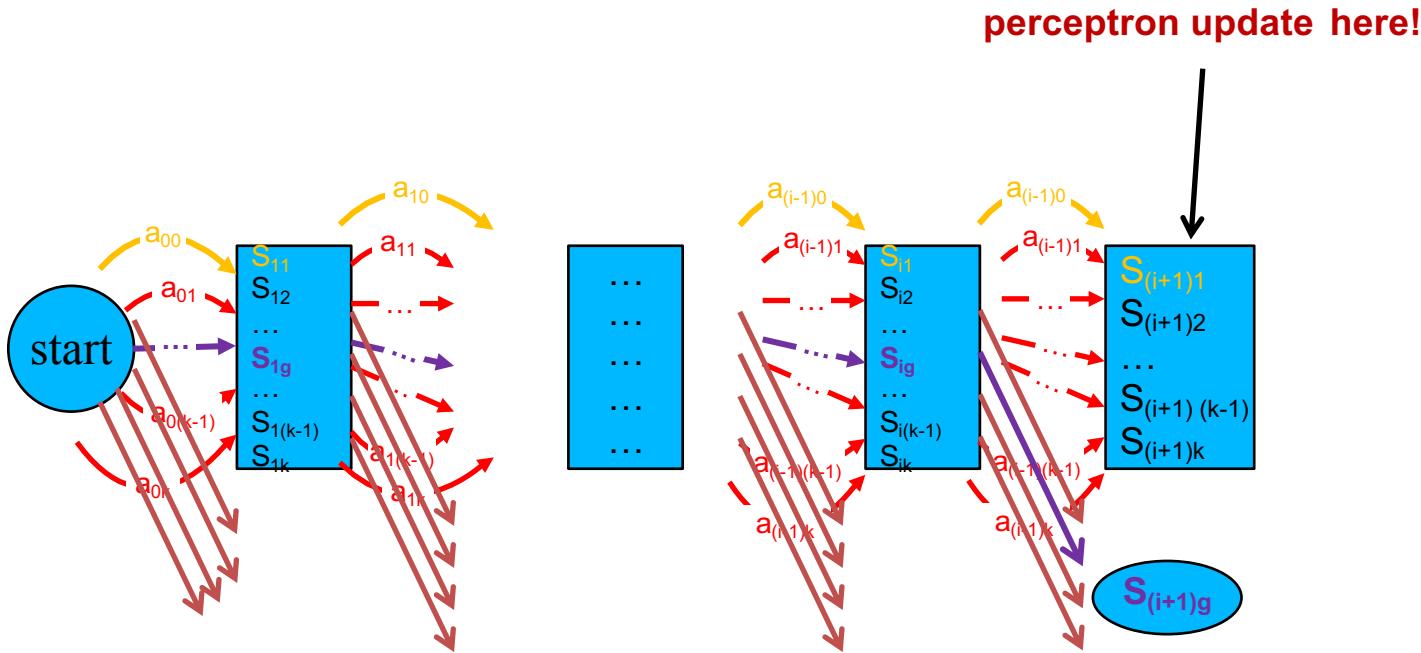
# Online learning



# Online learning



# Online learning



# Online learning

**Inputs:** training examples ( $x_i, y_i = \{S_0^i S_1^i \cdots S_m^i\}$  is a state sequence) $_1^N$

**Initialization:** set  $\vec{w} = 0$

**Algorithm:**

**for**  $r = 1 \cdots P, i = 1 \cdots N$  **do**

$candidates \leftarrow \{S_0^i\}$

$agenda \leftarrow \text{CLEAR}(agenda)$

**for**  $k = 1 \cdots m, m$  corresponds to a specific training example. **do**

**for** each candidate in  $candidates$  **do**

$agenda \leftarrow \text{INSERT}(\text{EXPAND}(candidate), agenda)$

$candidates \leftarrow \text{TOP-B}(agenda, B)$

$best \leftarrow \text{TOP}(agenda)$

**if**  $S_k^i$  is not in  $candidates$  or ( $best \neq S_m^i$  and  $k$  equals  $m$ ) **then**

$\vec{w} = \vec{w} + \Phi(S_k^i) - \Phi(best)$

**end if**

**end for**

**end for**

**end for**

**Output:**  $\vec{w}$

# The main strengths

- Fast
- Arbitrary nonlocal features
- Learning fixes search

# State-of-the-art results

- Chinese
  - Word segmentation
    - Yue Zhang and Stephen Clark. Chinese Segmentation Using a Word-Based Perceptron Algorithm. In proceedings of ACL 2007. Prague, Czech Republic. June.

# State-of-the-art results

- Chinese
  - Joint segmentation and POS-tagging
    - Yue Zhang and Stephen Clark. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In proceedings of ACL 2008. Ohio, USA. June.
    - Yue Zhang and Stephen Clark. A Fast Decoder for Joint Word Segmentation and POS-tagging Using a Single Discriminative Model. In proceedings of EMNLP 2010. Massachusetts, USA. October.

# State-of-the-art results

- Chinese
  - Joint segmentation, POS-tagging and chunking
    - Chen Lyu, Yue Zhang and Donghong Ji. Joint Word Segmentation, POS-Tagging and Syntactic Chunking. In Proceedings of the AAAI 2016, Phoenix, Arizona, USA, February

# State-of-the-art results

- Chinese
  - Joint segmentation, POS-tagging and dependency parsing
  - Meishan Zhang, Yue Zhang, Wanxiang Che and Ting Liu. Character-Level Chinese Dependency Parsing. In Proceedings of ACL 2014. Baltimore, USA, June.

# State-of-the-art results

- Chinese
  - Joint segmentation, POS-tagging and constituent parsing
  - Meishan Zhang, Yue Zhang, Wanxiang Che and Ting Liu. Chinese Parsing Exploiting Characters. In proceedings of ACL 2013. Sophia, Bulgaria. August.

# State-of-the-art results

- Chinese
  - Joint segmentation, POS-tagging and normalization
    - Tao Qian, Yue Zhang, Meishan Zhang and Donghong Ji. A Transition-based Model for Joint Segmentation, POS-tagging and Normalization. In proceedings of EMNLP 2015, Lisboa, Portugal, September.

# State-of-the-art results

- All Languages
  - Constituent parsing
    - Yue Zhang and Stephen Clark. Transition-Based Parsing of the Chinese Treebank Using a Global Discriminative Model. In proceedings of IWPT 2009. Paris, France. October.
    - Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang and Jingbo Zhu. Fast and Accurate Shift-Reduce Constituent Parsing. In proceedings of ACL 2013. Sophia, Bulgaria. August.

# State-of-the-art results

- All Languages
  - Dependency parsing
    - Yue Zhang and Stephen Clark. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In proceedings of ACL 2008. Ohio, USA. June.
    - Yue Zhang and Joakim Nivre. Transition-Based Dependency Parsing with Rich Non-Local Features. In proceedings of ACL 2011, short papers. Portland, USA. June.
    - Yue Zhang and Joakim Nivre. Analyzing the Effect of Global Learning and Beam-Search for Transition-Based Dependency Parsing. In proceedings of COLING 2012, posters. Mumbai, India. December.
    - Ji Ma, Yue Zhang and Jingbo Zhu. Punctuation Processing for Projective Dependency Parsing. In Proceedings of ACL 2014. Baltimore, USA, June.

# State-of-the-art results

- All Languages
  - CCG parsing
    - Yue Zhang and Stephen Clark. Shift-Reduce CCG Parsing. In proceedings of ACL 2011. Portland, USA. June.
    - Wenduan Xu, Stephen Clark and Yue Zhang. Shift-Reduce CCG Parsing with a Dependency Model. In Proceedings of ACL 2014. Baltimore, USA, June.

# State-of-the-art results

- All Languages
  - Natural language synthesis
    - Yijia Liu, Yue Zhang, Wanxiang Che and Bing Qin. Transition-Based Syntactic Linearization. In Proceedings of NAACL 2015, Denver, Colorado, USA, May.
    - Jiangming Liu and Yue Zhang. An Empirical Comparison Between N-gram and Syntactic Language Models for Word Ordering. In proceedings of EMNLP 2015, Lisboa, Portugal, September.
    - Ratish Puduppully, Yue Zhang and Manish Srivastava. Transition-Based Syntactic Linearization with Lookahead Features. In Proceedings of the NAACL 2016, San Diego, USA, June.

# State-of-the-art results

- All Languages
  - Joint morphological generation and text linearization
    - Linfeng Song, Yue Zhang, Kai Song and Qun Liu. Joint Morphological Generation and Syntactic Linearization. In Proceedings of AAAI 2014. Quebec City, Canada, July.

# State-of-the-art results

- All Languages
  - Joint entity and relation extraction
    - Fei Li, Yue Zhang, Meishan Zhang and Donghong Ji. Joint Models for Extracting Adverse Drug Events from Biomedical Text. In Proceedings of IJCAI 2016. New York City, USA, July.

# Part 5.2: A Neural Network Version

# Neural Network Model

- Use NN to substitute perceptron
- Why?
  - Better non-linear power
  - Unsupervised word embeddings
  - Automatic feature combination
  - Shown useful in greedy models

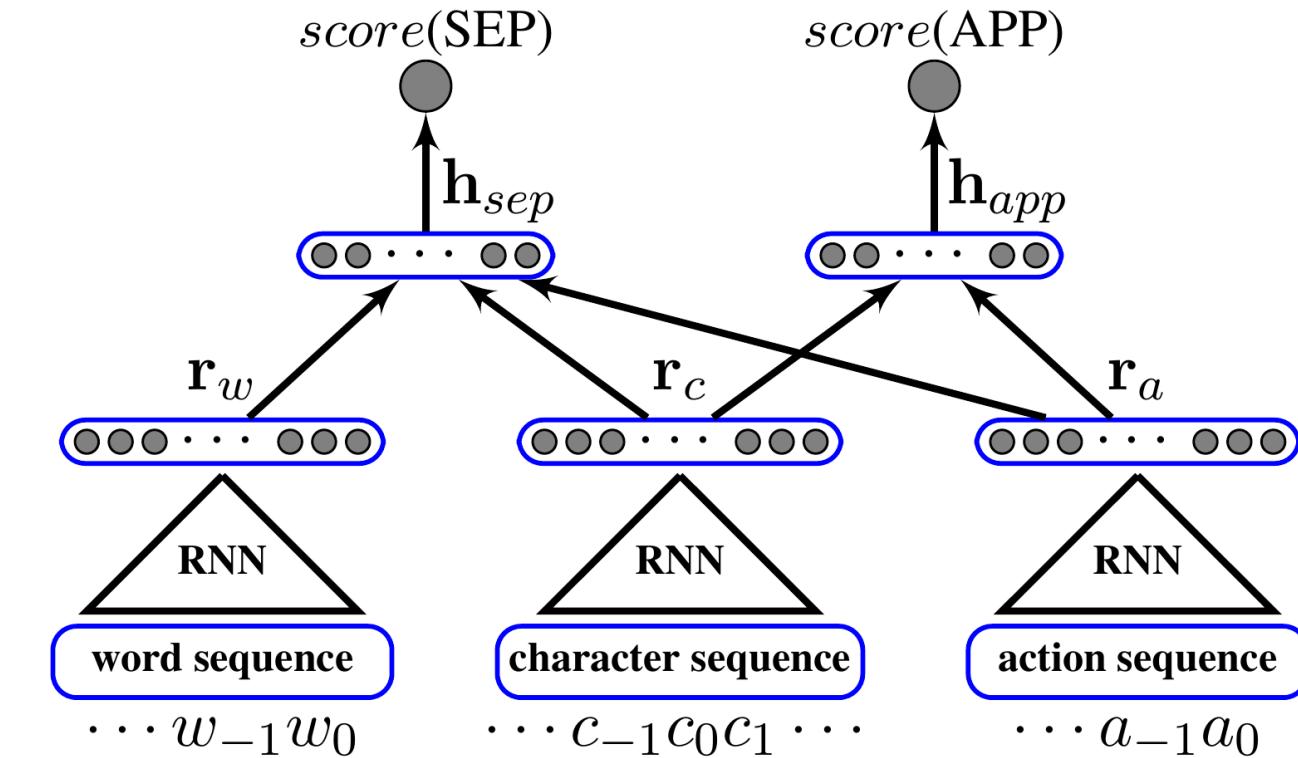
# Word segmentation

step	action	buffer( $\cdots w_{-1}w_0$ )	queue( $c_0c_1 \cdots$ )
0	-	$\phi$	中 国 ...
1	<i>SEP</i>	中	国 外 ...
2	<i>APP</i>	中国	外 企 ...
3	<i>SEP</i>	中国 外	企 业 ...
4	<i>APP</i>	中国 外企	业 务 ...
5	<i>SEP</i>	中国 外企 业	务 发 ...
6	<i>APP</i>	中国 外企 业务	发 展 ...
7	<i>SEP</i>	... 业务 发	展 迅 速
8	<i>APP</i>	... 业务 发展	迅 速
9	<i>SEP</i>	... 发展 迅	速
10	<i>APP</i>	... 发展 迅速	$\phi$

# Word segmentation

Feature templates	Action
$c_{-1}c_0$	$APP, SEP$
$w_{-1}, w_{-1}w_{-2}, w_{-1}c_0, w_{-2}len(w_{-1})$	
$start(w_{-1})c_0, end(w_{-1})c_0$	
$start(w_{-1})end(w_{-1}), end(w_{-2})end(w_{-1})$	$SEP$
$w_{-2}len(w_{-1}), len(w_{-2})w_{-1}$	
$w_{-1}$ , where $len(w_{-1}) = 1$	

# Word segmentation



# Word segmentation

Models	P	R	F
word-based models			
discrete	95.29	95.26	95.28
neural	95.34	94.69	95.01
combined	<b>96.11</b>	<b>95.79</b>	<b>95.95</b>
character-based models			
discrete	95.38	95.12	95.25
neural	94.59	94.92	94.76
combined	95.63	95.60	95.61
other models			
Zhang et al. (2014)	N/A	N/A	95.71
Wang et al. (2011)	95.83	95.75	95.79
Zhang and Clark (2011)	95.46	94.78	95.13

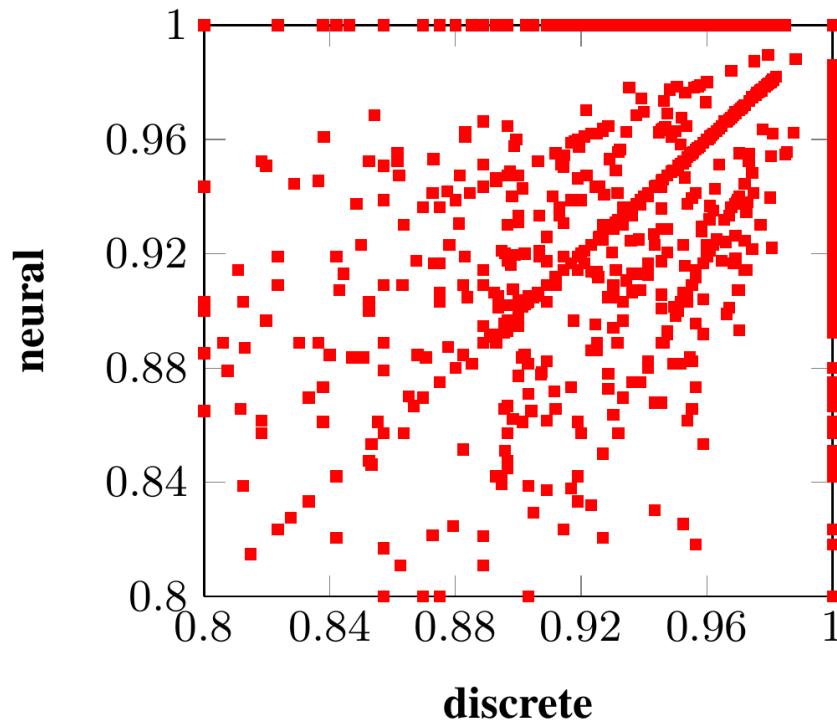
Main results on CTB60 test dataset

# Word segmentation

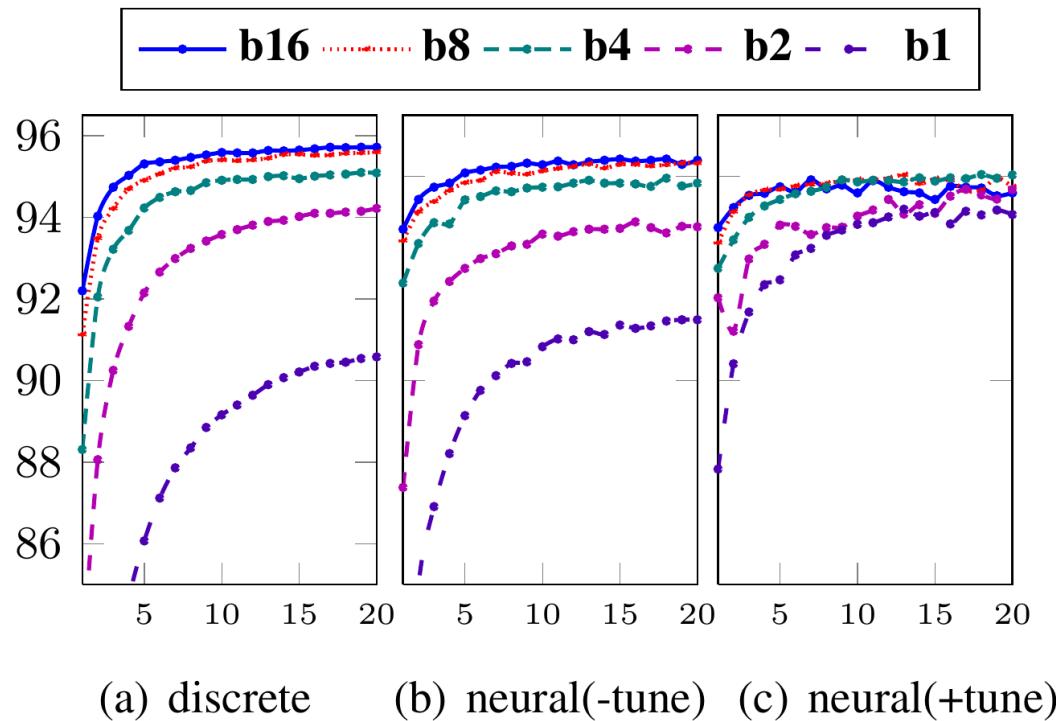
Models	PKU	MSR
our word-based models		
discrete	95.1	97.3
neural	95.1	97.0
combined	95.7	<b>97.7</b>
character-based models		
discrete	94.9	96.8
neural	94.4	97.2
combined	95.4	97.2
other models		
Cai and Zhao (2016)	95.5	96.5
Ma and Hinrichs (2015)	95.1	96.6
Pei et al. (2014)	95.2	97.2
Zhang et al. (2013a)	<b>96.1</b>	97.5
Sun et al. (2012)	95.4	97.4
Zhang and Clark (2011)	95.1	97.1
Sun (2010)	95.2	96.9
Sun et al. (2009)	95.2	97.3

Main results on PKU and MSR test dataset

# Word segmentation



# Word segmentation



(a) discrete      (b) neural(-tune)    (c) neural(+tune)

# Word segmentation

- Cai and Zhao (2016) presents a similar idea

# Dependency Parsing

- Zhang & Nivre (2011)

$$y = \arg \max_{y' \in \text{GEN}(x)} \text{score}(y')$$

$$\text{score}(y) = \sum_{a \in y} \theta \cdot \Phi(a)$$

# Dependency Parsing

- Chen and Manning (2014)

$$h = (W_1x + b_1)^3$$

$$p = softmax(o)$$

$$o = W_2h$$

# Dependency Parsing

- What does not work

$$s(y) = \sum_{a \in y} \log p_a$$

$$L(\theta) = \max(0, \delta - s(y_g) + s(y_p)) + \frac{\lambda}{2} \parallel \theta \parallel^2$$

# Dependency Parsing

- Sentence-level log likelihood

$$p(y_i \mid x, \theta) = \frac{e^{f(x, \theta)_i}}{\sum_{y_j \in \text{GEN}(x)} e^{f(x, \theta)_j}}$$

$$f(x, \theta)_i = \sum_{a_k \in y_i} o(x, y_i, k, a_k)$$

# Dependency Parsing

- Contrastive Estimation

$$\begin{aligned} L(\theta) &= - \sum_{(x_i, y_i) \in (X, Y)} \log p(y_i \mid x_i, \theta) \\ &= - \sum_{(x_i, y_i) \in (X, Y)} \log \frac{e^{f(x_i, \theta)_i}}{Z(x_i, \theta)} \\ &= \sum_{(x_i, y_i) \in (X, Y)} \log Z(x_i, \theta) - f(x_i, \theta)_i \end{aligned}$$

$$Z(x, \theta) = \sum_{y_j \in \text{GEN}(x)} e^{f(x, \theta)_j}$$

Zhou, H., Zhang, Y., Huang, S., & Chen, J. (2015). A Neural Probabilistic Structured-Prediction Model for Transition-Based Dependency Parsing. ACL.

# Dependency Parsing

- Contrastive Estimation

$$\begin{aligned} L'(\theta) &= - \sum_{(x_i, y_i) \in (X, Y)} \log p'(y_i \mid x_i, \theta) \\ &= - \sum_{(x_i, y_i) \in (X, Y)} \log \frac{e^{f(x_i, \theta)_i}}{Z'(x_i, \theta)} \\ &= \sum_{(x_i, y_i) \in (X, Y)} \log Z'(x_i, \theta) - f(x_i, \theta)_i \\ Z'(x, \theta) &= \sum_{y_j \in \text{BEAM}(x)} e^{f(x, \theta)_j} \end{aligned}$$

# Dependency Parsing

- Results

Description	UAS	
Baseline	91.63	
	structured	greedy
beam = 1	74.90	91.63
beam = 4	84.64	91.92
beam = 16	91.53	91.90
beam = 64	93.12	91.84
beam = 100	93.23	91.81

# Dependency Parsing

- Results

Description	UAS
greedy neural parser	91.47
ranking model	89.08
beam contrastive learning	93.28

# Dependency Parsing

- Results

System	UAS	LAS	Speed	
baseline greedy parser	91.47	90.43	0.001	
Huang and Sagae (2010)	92.10		0.04	
Zhang and Nivre (2011)	92.90	91.80	0.03	
Choi and McCallum (2013)	92.96	91.93	0.009	
Ma et al. (2014)	93.06			
Bohnet and Nivre (2012)†‡	93.67	92.68	0.4	
Suzuki et al. (2009)†	93.79			
Koo et al. (2008)†	93.16			
Chen et al. (2014)†	93.77			
beam size				
training	decoding			
100	100	<b>93.28</b>	<b>92.35</b>	0.07
100	64	93.20	92.27	0.04
100	16	92.40	91.95	0.01

# Google

- Andor et al. follows this method
  - Offers theorem
  - Tries more tasks
  - Get better results

# Google

- Dependency parsing

Method	WSJ		Union-News		Union-Web		Union-QTB	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Martins et al. (2013)*	92.89	90.55	93.10	91.13	88.23	85.04	94.21	91.54
Zhang and McDonald (2014)*	93.22	91.02	93.32	91.48	88.65	85.59	93.37	90.69
Weiss et al. (2015)	93.99	92.05	93.91	92.25	89.29	86.44	94.17	92.06
Alberti et al. (2015)	94.23	92.36	94.10	92.55	89.55	86.85	94.74	93.04
Our Local (B=1)	92.95	91.02	93.11	91.46	88.42	85.58	92.49	90.38
Our Local (B=32)	93.59	91.70	93.65	92.03	88.96	86.17	93.22	91.17
Our Global (B=32)	<b>94.61</b>	<b>92.79</b>	<b>94.44</b>	<b>92.93</b>	<b>90.17</b>	<b>87.54</b>	<b>95.40</b>	<b>93.64</b>
Parsey McParseface (B=8)	-	-	94.15	92.51	89.08	86.29	94.77	93.17

# Google

- Dependency parsing

Method	Catalan		Chinese		Czech		English		German		Japanese		Spanish	
	UAS	LAS												
Best Shared Task Result	-	87.86	-	79.17	-	80.38	-	89.88	-	87.48	-	92.57	-	87.64
Ballesteros et al. (2015)	90.22	86.42	80.64	76.52	79.87	73.62	90.56	88.01	88.83	86.10	93.47	92.55	90.38	86.59
Zhang and McDonald (2014)	91.41	87.91	82.87	78.57	86.62	80.59	92.69	90.01	89.88	87.38	92.82	91.87	90.82	87.34
Lei et al. (2014)	91.33	87.22	81.67	76.71	88.76	81.77	92.75	90.00	90.81	87.81	<b>94.04</b>	91.84	91.16	87.38
Bohnet and Nivre (2012)	92.44	89.60	82.52	78.51	88.82	83.73	92.87	90.60	<b>91.37</b>	<b>89.38</b>	93.67	92.63	92.24	89.60
Alberti et al. (2015)	92.31	89.17	83.57	79.90	88.45	83.57	92.70	90.56	90.58	88.20	93.99	<b>93.10</b>	92.26	89.33
Our Local (B=1)	91.24	88.21	81.29	77.29	85.78	80.63	91.44	89.29	89.12	86.95	93.71	92.85	91.01	88.14
Our Local (B=16)	91.91	88.93	82.22	78.26	86.25	81.28	92.16	90.05	89.53	87.4	93.61	92.74	91.64	88.88
Our Global (B=16)	<b>92.67</b>	<b>89.83</b>	<b>84.72</b>	<b>80.85</b>	<b>88.94</b>	<b>84.56</b>	<b>93.22</b>	<b>91.23</b>	90.91	89.15	93.65	92.84	<b>92.62</b>	<b>89.95</b>

# Google

- POS-tagging

Method	En	En-Union			CoNLL '09						Avg	
	WSJ	News	Web	QTB	Ca	Ch	Cz	En	Ge	Ja	Sp	
Linear CRF	97.17	97.60	94.58	96.04	98.81	94.45	98.90	97.50	97.14	97.90	98.79	97.17
Ling et al. (2015)	<b>97.78</b>	97.44	94.03	96.18	98.77	94.38	99.00	97.60	<b>97.84</b>	97.06	98.71	97.16
Our Local (B=1)	97.44	97.66	94.46	96.59	98.91	94.56	98.96	97.36	97.35	98.02	98.88	97.29
Our Local (B=8)	97.45	97.69	94.46	96.64	98.88	94.56	98.96	97.40	97.35	98.02	98.89	97.30
Our Global (B=8)	97.44	<b>97.77</b>	<b>94.80</b>	<b>96.86</b>	<b>99.03</b>	<b>94.72</b>	<b>99.02</b>	<b>97.65</b>	97.52	<b>98.37</b>	<b>98.97</b>	<b>97.47</b>
Parsey McParseface	-	97.52	94.24	96.45	-	-	-	-	-	-	-	-

# Google

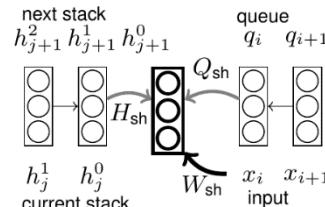
- Compression

Method	Generated corpus		Human eval	
	A	F1	read	info
Filippova et al. (2015)	<b>35.36</b>	<b>82.83</b>	4.66	4.03
Automatic	-	-	4.31	3.77
Our Local (B=1)	30.51	78.72	4.58	4.03
Our Local (B=8)	31.19	75.69	-	-
Our Global (B=8)	35.16	81.41	<b>4.67</b>	<b>4.07</b>

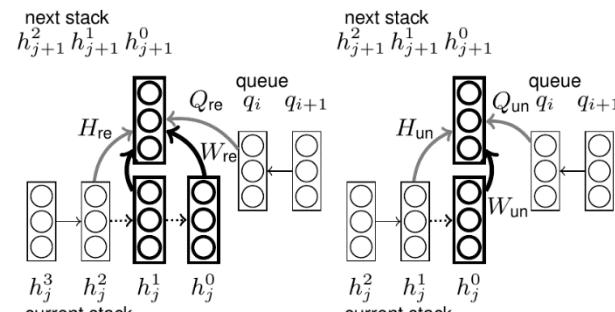
## Part 5.3: Similar methods by others

# Other methods ( I )

- Constituent parsing



(a) shift- $X$  action



(b) reduce- $X$  action

(c) unary- $X$  action

# Other methods ( I )

- Update at max-violation

$$j^* = \arg \min_j \left\{ \rho_{\boldsymbol{\theta}}(y_0^j) - \max_{\mathbf{d} \in B_j} \rho_{\boldsymbol{\theta}}(\mathbf{d}) \right\}$$

- Using expected loss from all violations

$$L(\mathbf{w}, \mathbf{y}; \mathbf{B}, \boldsymbol{\theta}) = \max \left\{ 0, 1 - \rho_{\boldsymbol{\theta}}(y_0^{j^*}) + \mathbb{E}_{\tilde{B}_{j^*}} [\rho_{\boldsymbol{\theta}}] \right\}$$

$$\tilde{B}_{j^*} = \left\{ \mathbf{d} \in B_{j^*} \mid \rho_{\boldsymbol{\theta}}(\mathbf{d}) > \rho_{\boldsymbol{\theta}}(y_0^{j^*}) \right\}$$

$$p_{\boldsymbol{\theta}}(\mathbf{d}) = \frac{\exp(\rho_{\boldsymbol{\theta}}(\mathbf{d}))}{\sum_{\mathbf{d}' \in \tilde{B}_{j^*}} \exp(\rho_{\boldsymbol{\theta}}(\mathbf{d}'))}$$

$$\mathbb{E}_{\tilde{B}_{j^*}} [\rho_{\boldsymbol{\theta}}] = \sum_{\mathbf{d} \in \tilde{B}_{j^*}} p_{\boldsymbol{\theta}}(\mathbf{d}) \rho_{\boldsymbol{\theta}}(\mathbf{d}).$$

# Other methods ( I )

parser	test
Collins (Collins, 1997)	87.8
Berkeley (Petrov and Klein, 2007)	90.1
SSN (Henderson, 2004)	90.1
ZPar (Zhu et al., 2013)	90.4
CVG (Socher et al., 2013)	90.4
Charniak-R (Charniak and Johnson, 2005)	<b>91.0</b>
This work: TNCP	90.7

# Other methods ( I )

parser	test
ZPar (Zhu et al., 2013)	83.2
Berkeley (Petrov and Klein, 2007)	83.3
Joint (Wang and Xue, 2014)	<b>84.9</b>
This work: TNCP	84.3

# Other methods ( II )

- CCG Parsing
- expected F1 training

$$\begin{aligned} J(\theta) &= -\mathbf{x}\mathbf{F1}(\theta) \\ &= - \sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\mathbf{F1}(\Delta_{y_i}, \Delta_{x_n}^G) \end{aligned}$$

$$p(y_i|\theta) = \frac{\exp\{\rho(y_i)\}}{\sum_{y \in \Lambda(x_n)} \exp\{\rho(y)\}}$$

# Other methods ( II )

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= - \sum_{y_i \in \Lambda(x_n)} \sum_{y_{ij} \in y_i} \frac{\partial J(\theta)}{\partial s_\theta(y_{ij})} \frac{\partial s_\theta(y_{ij})}{\partial \theta} \\ &= - \sum_{y_i \in \Lambda(x_n)} \sum_{y_{ij} \in y_i} \delta_{y_{ij}} \frac{\partial s_\theta(y_{ij})}{\partial \theta},\end{aligned}$$

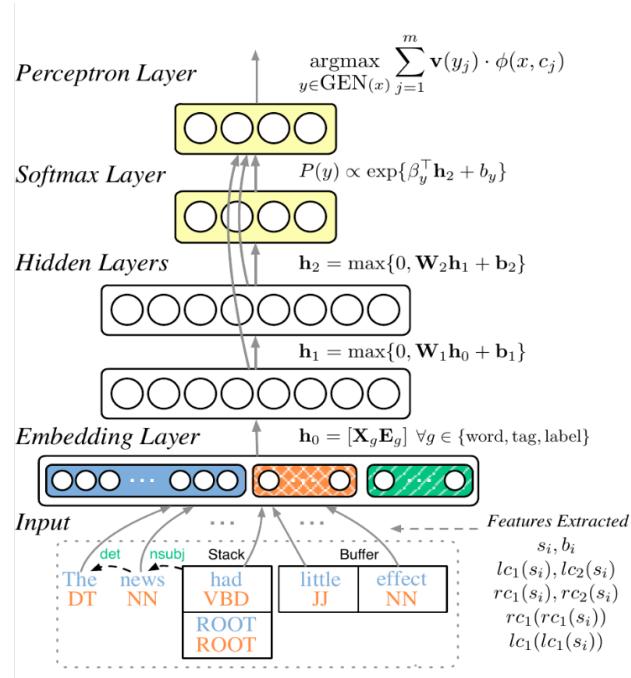
$$\begin{aligned}\delta_{y_{ij}} &= - \frac{\partial \text{xF1}(\theta)}{\partial s_\theta(y_{ij})} \\ &= - \frac{\partial(G(\theta)/Z(\theta))}{\partial s_\theta(y_{ij})} \\ &= \frac{G(\theta)Z'(\theta) - G'(\theta)Z(\theta)}{Z^2(\theta)} \\ &= \frac{\exp\{\rho(y_i)\}}{Z(\theta)} (\text{xF1}(\theta) - \text{F1}(\Delta_{y_i}, \Delta_{x_n}^G)) \frac{1}{s_\theta(y_{ij})} \\ &= p(y_i|\theta) (\text{xF1}(\theta) - \text{F1}(\Delta_{y_i}, \Delta_{x_n}^G)) \frac{1}{s_\theta(y_{ij})},\end{aligned}$$

# Other methods ( II )

Model	Section 00				Section 23				Speed
	LP	LR	LF	CAT	LP	LR	LF	CAT	
C&C (normal)	85.18	82.53	83.83	92.39	85.45	83.97	84.70	92.83	97.90
C&C (hybrid)	86.07	82.77	84.39	92.57	86.24	84.17	85.19	93.00	95.25
Zhang and Clark (2011) ( $b = 16$ )	87.15	82.95	85.00	92.77	87.43	83.61	85.48	93.12	-
Zhang and Clark (2011)* ( $b = 16$ )	86.76	83.15	84.92	92.64	87.04	84.14	85.56	92.95	49.54
Xu et al. (2014) ( $b = 128$ )	86.29	<b>84.09</b>	85.18	92.75	87.03	<b>85.08</b>	86.04	93.10	12.85
RNN-greedy ( $b = 1$ )	88.12	81.38	84.61	93.42	88.53	81.65	84.95	93.57	337.45
RNN-greedy ( $b = 6$ )	87.96	82.27	85.02	93.47	88.54	82.77	85.56	93.68	96.04
RNN-xF1 ( $b = 8$ )	<b>88.20</b>	83.40	<b>85.73</b>	<b>93.56</b>	<b>88.74</b>	84.22	<b>86.42</b>	<b>93.87</b>	67.65

# Other methods (III)

- Dependency parsing



## Other methods (*III*)

- Using Chen and Manning features for perceptron training
- Back-propagation pre-training

$$L(\Theta) = - \sum_j \log P(y_j | c_j, \Theta) + \lambda \sum_i \|\mathbf{W}_i\|_2^2$$

- Structured perceptron training

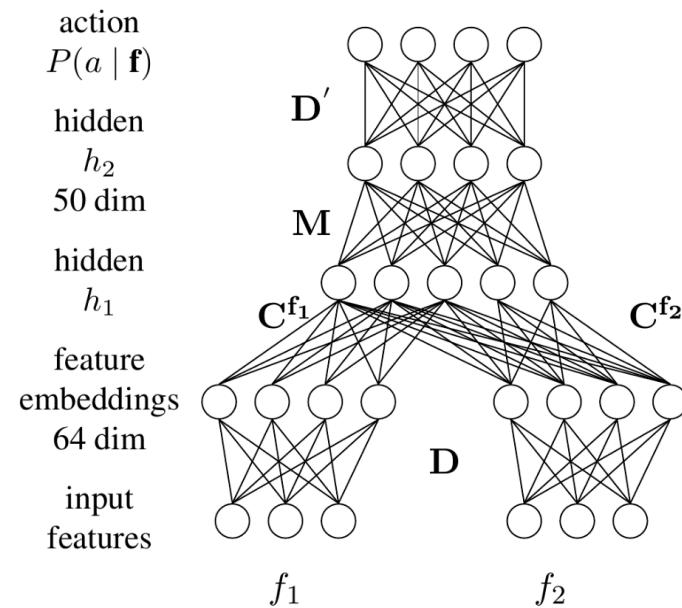
$(h_1, h_2, P(y))$

# Other methods (*III*)

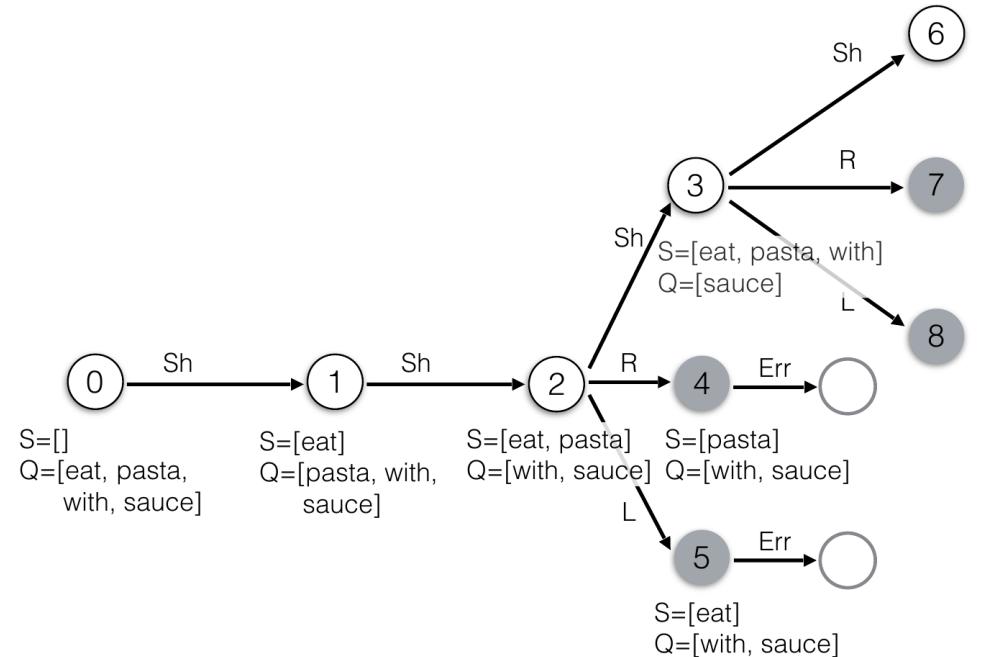
Method	UAS	LAS	Beam
<i>Graph-based</i>			
Bohnet (2010)	92.88	90.71	n/a
Martins et al. (2013)	92.89	90.55	n/a
Zhang and McDonald (2014)	93.22	91.02	n/a
<i>Transition-based</i>			
*Zhang and Nivre (2011)	93.00	90.95	32
Bohnet and Kuhn (2012)	93.27	91.19	40
Chen and Manning (2014)	91.80	89.60	1
S-LSTM (Dyer et al., 2015)	93.20	90.90	1
Our Greedy	93.19	91.18	1
Our Perceptron	<b>93.99</b>	<b>92.05</b>	8
<i>Tri-training</i>			
*Zhang and Nivre (2011)	92.92	90.88	32
Our Greedy	93.46	91.49	1
Our Perceptron	<b>94.26</b>	<b>92.41</b>	8

# Other methods (IV)

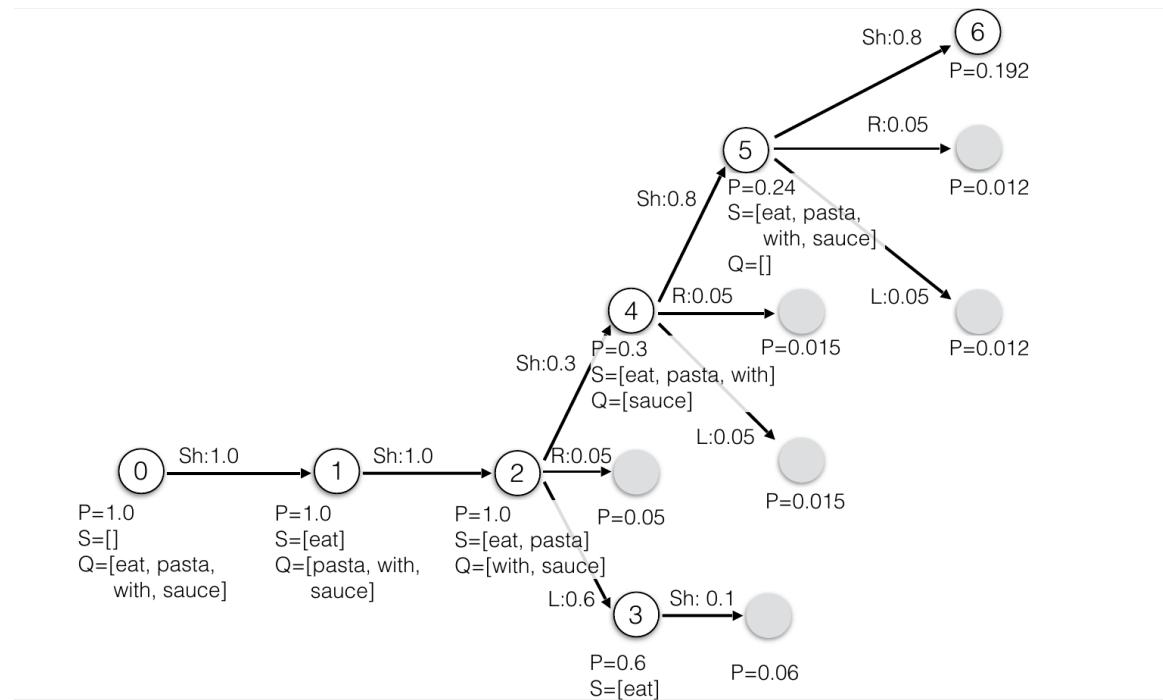
- Dependency parsing



# Other methods (IV)



# Other methods (IV)



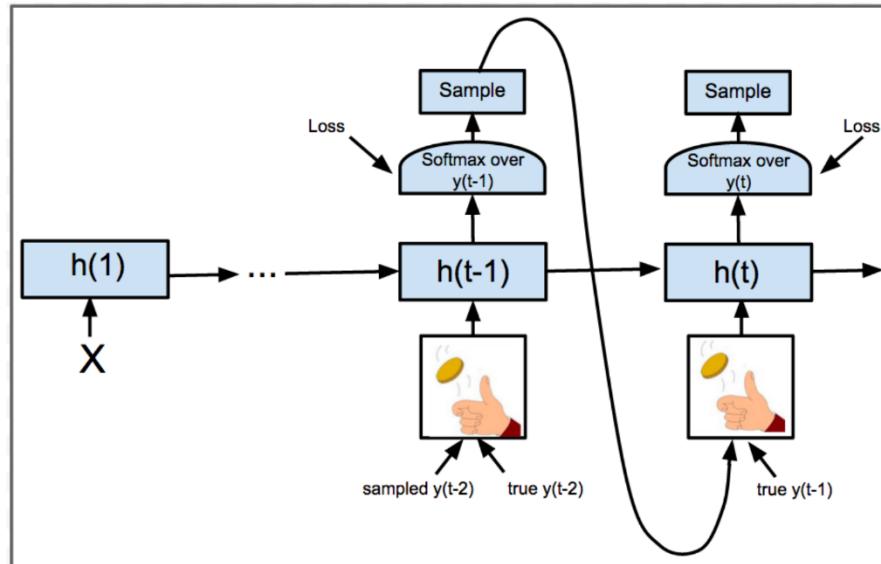
## Other methods (*IV*)

System	wsj23-S	wsj23-YM
<b>ErrSt-25-rand</b>	<b>92.17</b>	<b>92.16</b>
<b>ErrSt-25-pre*</b>	<b>93.61</b>	<b>93.21</b>
Chen & Manning*	91.8	–
Huang & Sagae	–	92.1
Zhang & Nivre	93.5	92.9
Weiss et al.*	93.99	–
Zhang & McDonald	93.71	93.57
Martins et al.	92.82	93.07
Koo et al. (dep2c)*	–	93.16

# Part 5.4: Beam-search Decoding for Sequence to Sequence Models

# Sequence to sequence ( I )

- Scheduled Sampling



Beam Search Inference

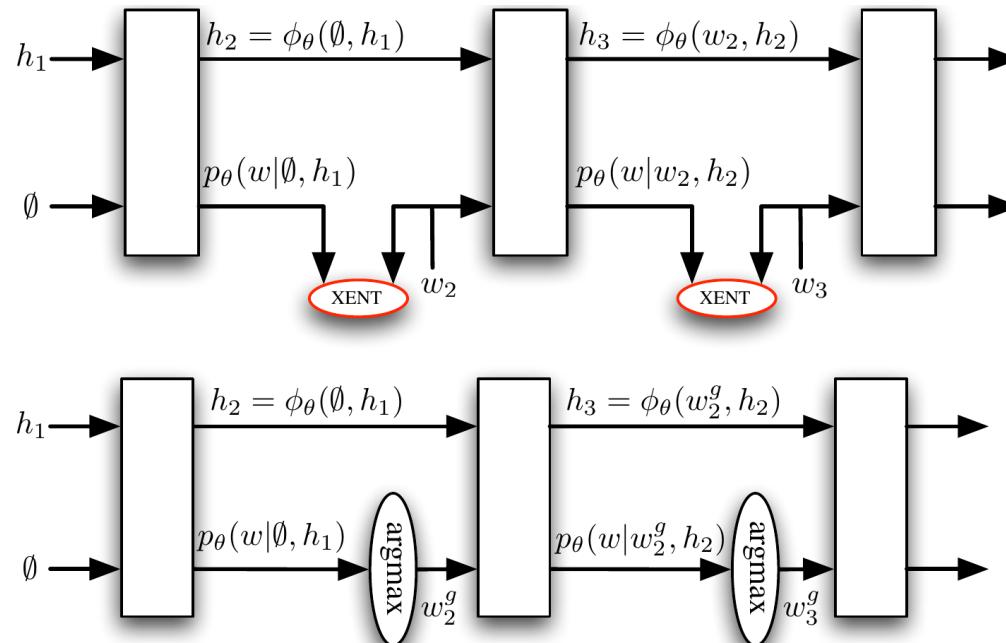
# Sequence to sequence ( I )

- Scheduled Sampling

Approach	F1
Baseline LSTM	86.54
Baseline LSTM with Dropout	87.0
Always Sampling	-
Scheduled Sampling	<b>88.08</b>
Scheduled Sampling with Dropout	<b>88.68</b>

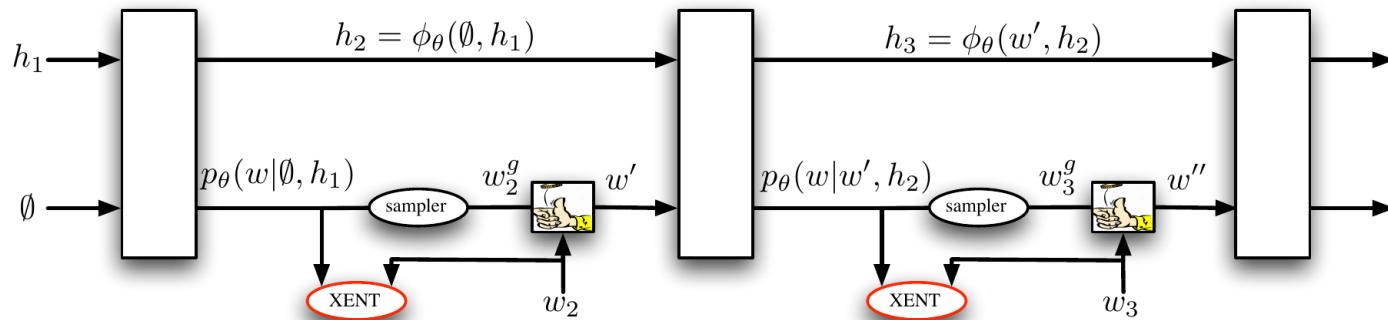
# Sequence to sequence ( II )

- Sequence-level training



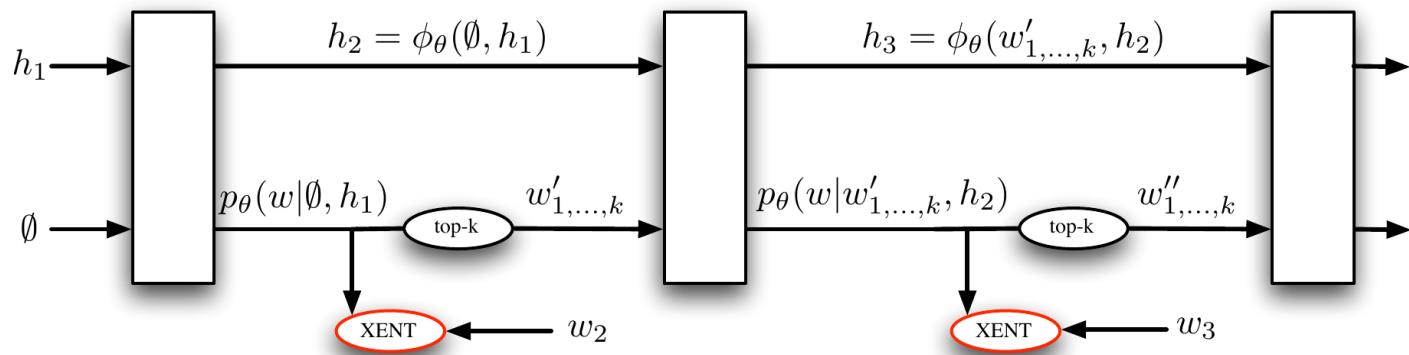
# Sequence to sequence ( II )

- Sequence-level training



# Sequence to sequence ( II )

- Sequence-level training



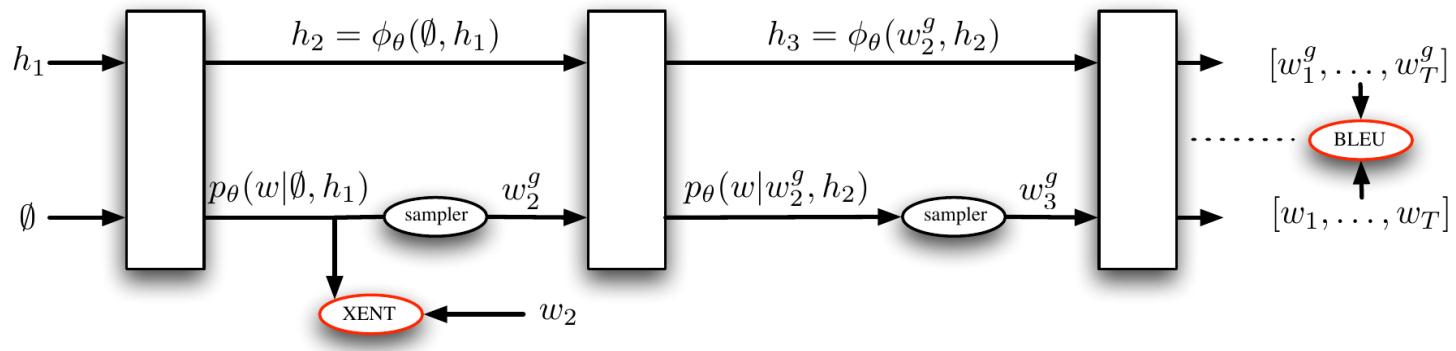
# Sequence to sequence ( II)

- Reinforce

$$L_\theta = - \sum_{w_1^g, \dots, w_T^g} p_\theta(w_1^g, \dots, w_T^g) r(w_1^g, \dots, w_T^g) = -\mathbb{E}_{[w_1^g, \dots, w_T^g] \sim p_\theta} r(w_1^g, \dots, w_T^g)$$

# Sequence to sequence ( II )

- Mixer



# Sequence to sequence ( II )

**Data:** a set of sequences with their corresponding context.

**Result:** RNN optimized for generation.

Initialize RNN at random and set  $N^{\text{XENT}}$ ,  $N^{\text{XE+R}}$  and  $\Delta$ ;

**for**  $s = T, 1, -\Delta$  **do**

**if**  $s == T$  **then**

        train RNN for  $N^{\text{XENT}}$  epochs using XENT only;

**else**

        train RNN for  $N^{\text{XE+R}}$  epochs. Use XENT loss in the first  $s$  steps, and REINFORCE (sampling from the model) in the remaining  $T - s$  steps;

**end**

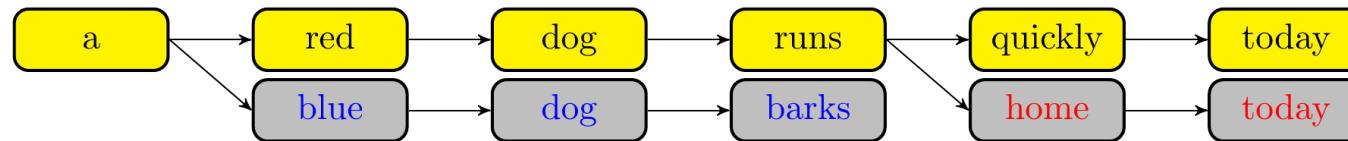
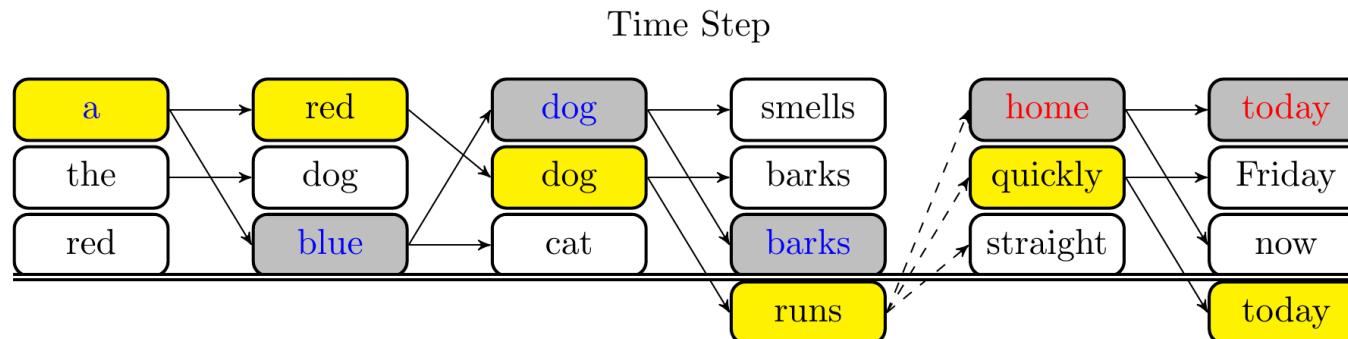
**end**

# Sequence to sequence ( $\Pi$ )

<i>TASK</i>	XENT	DAD	E2E	MIXER
<i>summarization</i>	13.01	12.18	12.78	<b>16.22</b>
<i>translation</i>	17.74	20.12	17.77	<b>20.73</b>
<i>image captioning</i>	27.8	28.16	26.42	<b>29.16</b>

# Sequence to sequence (III)

- Learning for Search



# Sequence to sequence (III)

$$\begin{aligned}\mathcal{L}(f) = \\ \sum_{t=1}^T \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - f(y_t, \mathbf{h}_{t-1}) + f(\hat{y}_t^{(K)}, \hat{\mathbf{h}}_{t-1}^{(K)}) \right]\end{aligned}$$

# Sequence to sequence (*III*)

- Need greedy pre-training

# Sequence to sequence (*III*)

- Curriculum beam increase

# Sequence to sequence (*III*)

	Word Ordering (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
seq2seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
ConBSO	<b>28.6</b>	<b>34.3</b>	<b>34.5</b>
LSTM-LM	15.4	-	26.8

# Sequence to sequence (III)

Dependency Parsing (UAS/LAS)			
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
seq2seq	<b>87.33/82.26</b>	88.53/84.16	88.66/84.33
BSO	86.91/82.11	<b>91.00/87.18</b>	<b>91.17/87.41</b>
ConBSO	85.11/79.32	<b>91.25/86.92</b>	<b>91.57/87.26</b>
Andor	93.17/91.18	-	-

# Sequence to sequence (*III*)

	Machine Translation (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
seq2seq	22.53	24.03	23.87
BSO, SB- $\Delta$	<b>23.83</b>	<b>26.36</b>	<b>25.48</b>
XENT	17.74	$\leq 20.5$	$\leq 20.5$
DAD	20.12	$\leq 22.5$	$\leq 23.0$
MIXER	20.73	-	$\leq 22.0$