Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology

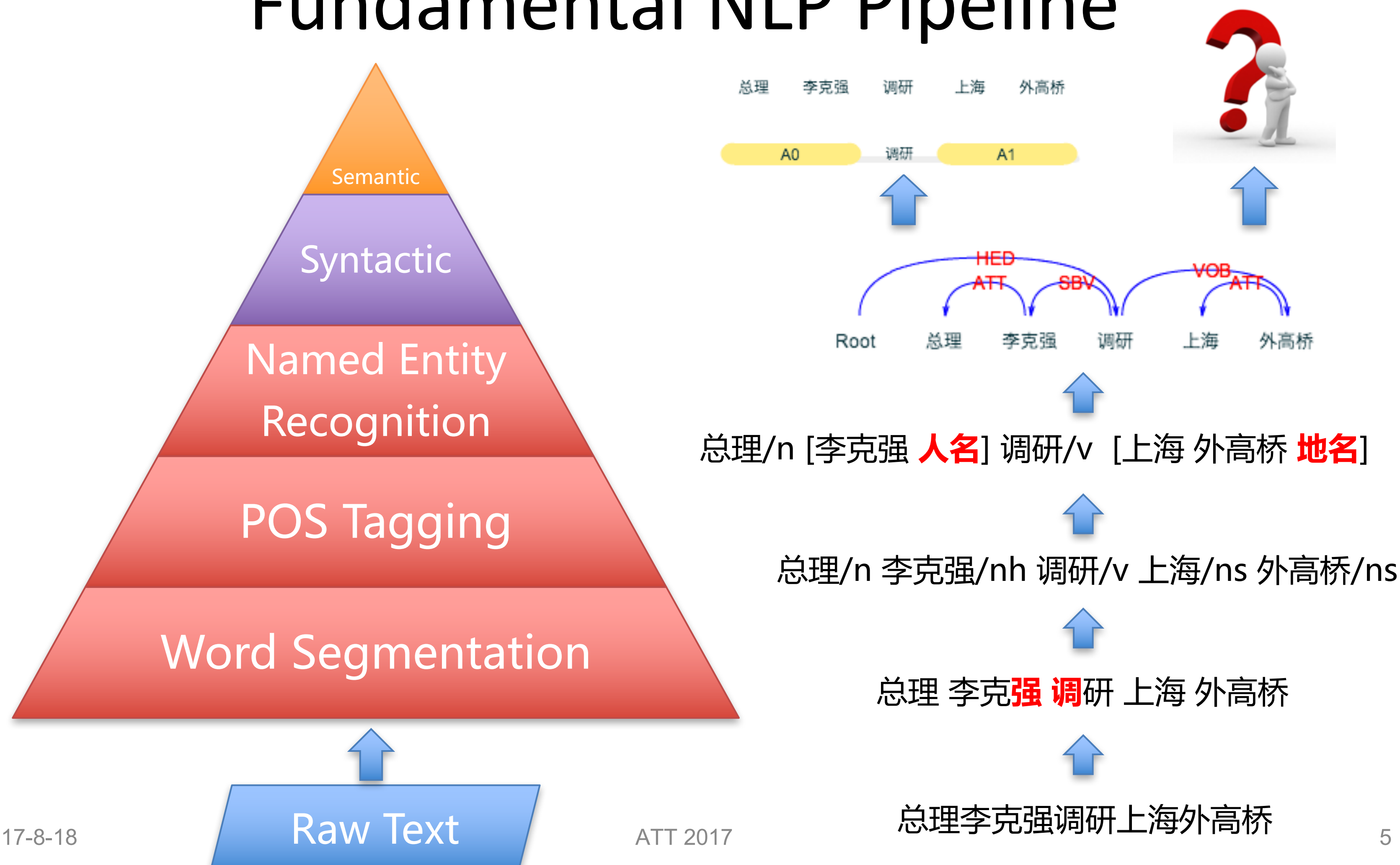
Outline

Timeline	Content
09:00-09:20	Task Introduction
09:20-09:50	Graph-based Methods
09:50-10:20	Transition-based Methods
10:20-10:40	Break
10:40-11:10	Neural Graph-based Methods
11:10-11:40	Neural Transition-based Methods
11:40-12:00	Applications

Part 1: Task Introduction

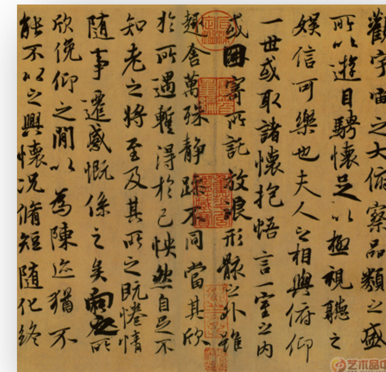
Part 1.1: Lexical, Syntactic and Semantic Analysis

Fundamental NLP Pipeline



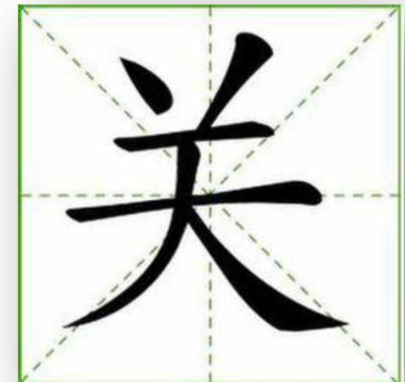
Word Segmentation

- 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

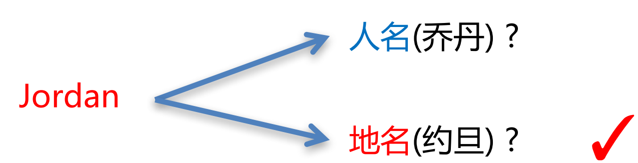
- A POS is a category of words which have similar grammatical properties
  - E.g. noun, verb, adjective
- POS tagging
  - Marking up a word in a text as a particular POS
  - based on both its definition and its context
- Ambiguities in POS Tagging
  - Time flies like an arrow.
  - 制服了敌人 vs. 穿着制服



Named Entity Recognition (NER)

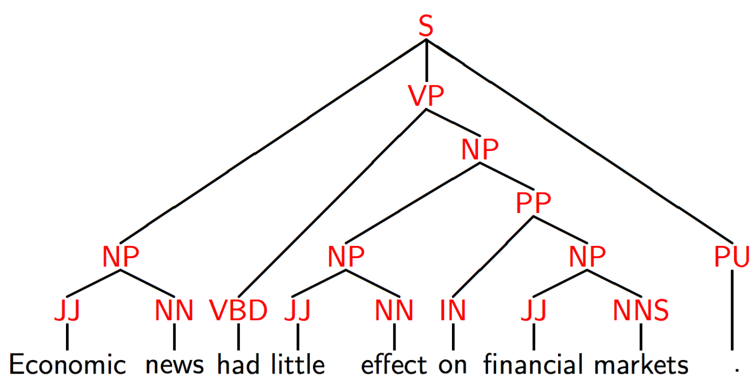
- Named Entities
  - Persons, locations, organizations, expressions of times, quantities, monetary values, percentages, etc.
- Locating and classifying named entities in text into pre-defined categories
- Ambiguities in NER

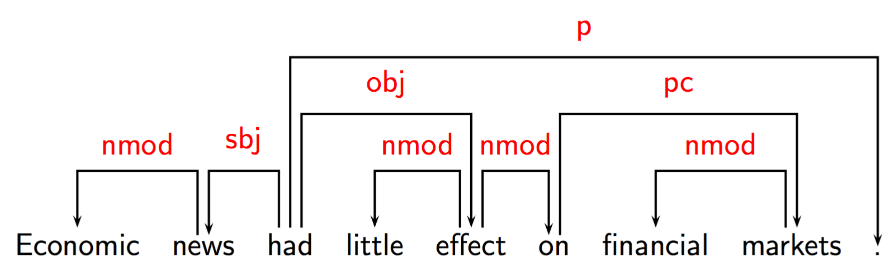
Kerry to visit Jordan, Israel Palestinian peace on agenda.



Syntactic Parsing

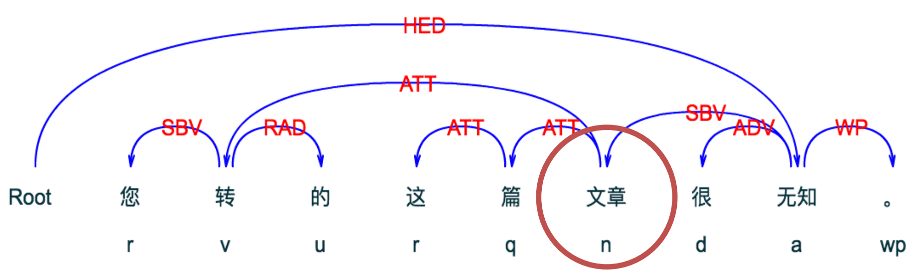
- Analyzing a natural language string conforming to the rules of a formal grammar, emphasizing subject, predicate, object, etc.
  - Constituency and Dependency Parsing

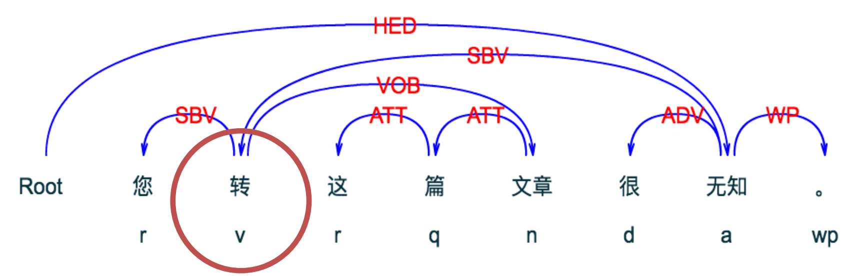




Dependency Parsing

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





http://ltp.ai/demo.html

Semantic Role Labeling

- Recognizing predicates and corresponding arguments

$\overbrace{\text{Yesterday}}^{\text{TEMP}}$, $\overbrace{\text{Kristina}}^{\text{HITTER}}$ hit $\overbrace{\text{Scott}}^{\text{THING HIT}}$ $\overbrace{\text{with a baseball}}^{\text{INSTRUMENT}}$

Scott was hit by Kristina yesterday with a baseball

Yesterday, Scott was hit with a baseball by Kristina

With a baseball, Kristina hit Scott yesterday

Yesterday Scott was hit by Kristina with a baseball

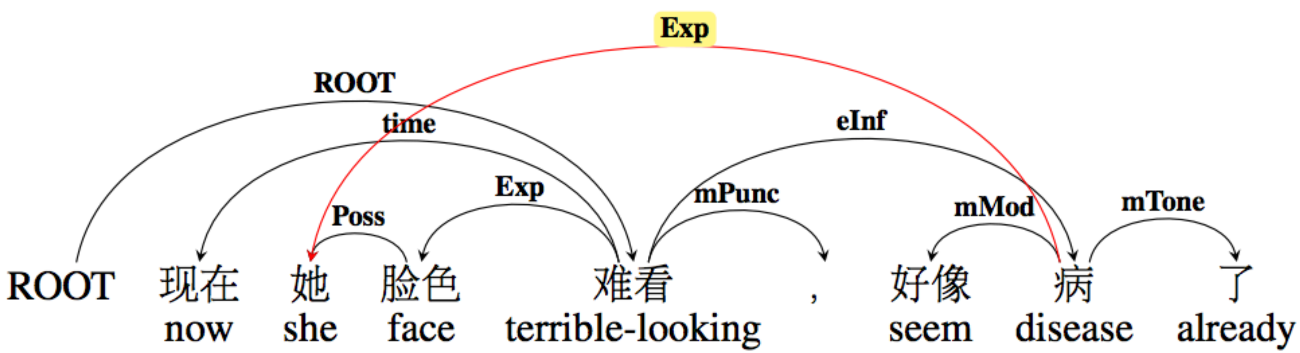
Kristina hit Scott with a baseball yesterday

Example from (Yih & Toutanova, 2006)

Semantic Role Labeling

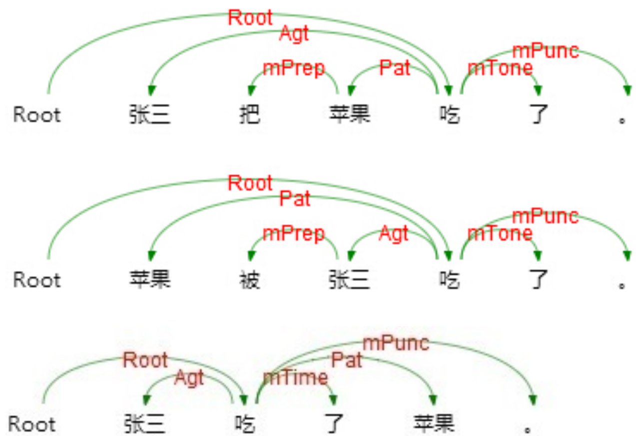
- Answer "Who did what to whom when and where"
  - Question Answering
    - Yesterday $_{\text{time}}$, Mary $_{\text{buyer}}$ bought a shirt $_{\text{bought thing}}$ from Tom $_{\text{seller}}$
    - Whom $_{\text{buyer}}$ did Tom $_{\text{seller}}$ sell a shirt $_{\text{bought thing}}$ to, yesterday $_{\text{time}}$
  - Information Extraction
  - ……

Semantic Dependency Graph

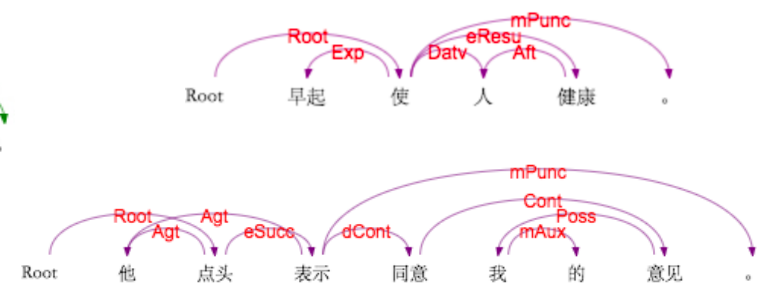


http://www.ltp-cloud.com/intro/#sdp_how

Semantic Dependency Parsing



语义依存树



语义依存图

https://www.cs.york.ac.uk/semeval-2012/task5.html

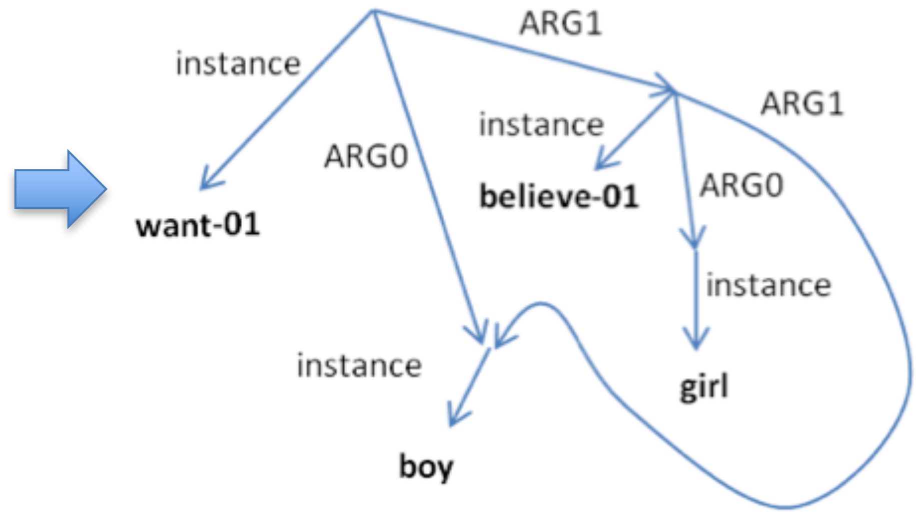
Wanxiang Che, Meishan Zhang, Yanqiu Shao, Ting Liu. **SemEval-2012 Task 5**: Chinese Semantic Dependency Parsing.

http://alt.qcri.org/semeval2016/task9/

Wanxiang Che, Yu Ding, Yanqiu Shao, Ting Liu. **SemEval-2016 Task 9**: Chinese Semantic Dependency Parsing.

Abstract Meaning Representation (AMR)

The boy wants the girl to believe him.
The boy wants to be believed by the girl.
The boy has a desire to be believed by the girl.
The boy's desire is for the girl to believe him.
The boy is desirous of the girl believing him.



http://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf

Combinatory Categorial Grammars (CCG)

$$
\begin{array}{ccc}
\text{CCG} & \text{is} & \text{fun} \\
\hline
NP & S\backslash NP/ADJ & ADJ \\
CCG & \lambda f.\lambda x.f(x) & \lambda x.fun(x) \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP \quad >} \\
 & \multicolumn{2}{c}{\lambda x.fun(x)} \\
\hline
\multicolumn{3}{c}{S \quad <} \\
\multicolumn{3}{c}{fun(CCG)}
\end{array}
$$

- CCG Lexical Entries
  - Pair words and phrases with meaning by a CCG category

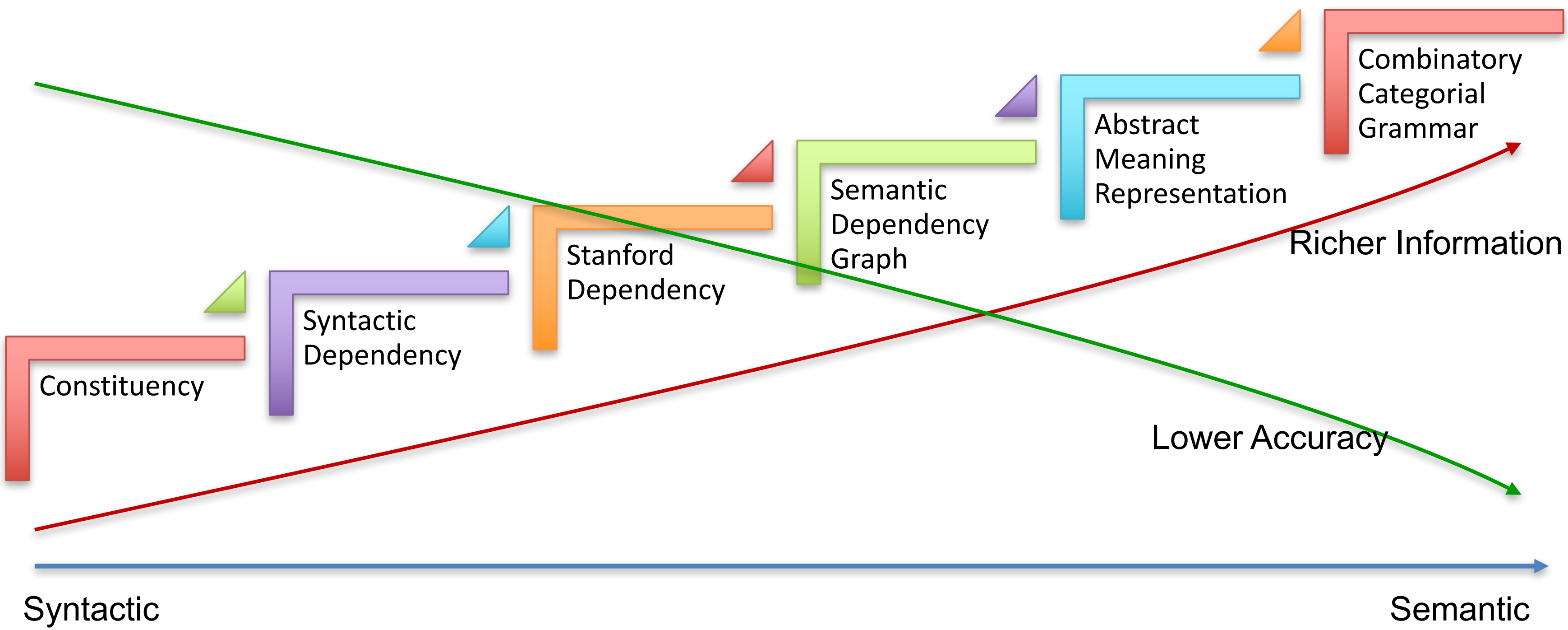
$\text{fun} \vdash ADJ : \lambda x.fun(x)$

Natural Language — CCG Category

- CCG Categories
  - Basic building block
  - Capture syntactic and semantic information jointly

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

Grammar


Constituency
Syntactic Dependency
Stanford Dependency
Semantic Dependency Graph
Abstract Meaning Representation
Combinatory Categorial Grammar
Richer Information
Lower Accuracy
Syntactic
Semantic

Part 1.2: Structured Prediction

Structured Prediction

- Predicting structured objects, rather than scalar discrete or real values
- Outputs are influenced each other
- Categories
  - Sequence segmentation
  - Sequence labeling / Tagging
  - Parsing

Sequence Segmentation

- Break a sequence into contiguous parts
- For example: Word Segmentation
  - Input
    - 严守一把手机关了
  - Output
    - 严守一/ 把/ 手机/ 关/ 了/
- More examples:
  - Sentence segmentation (a post-processing stage for speech transcription)
  - Paragraph segmentation

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

Word 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 $_{\text{time}}$], [Mary $_{\text{buyer}}$] bought/pred [a shirt $_{\text{bought thing}}$] from [Tom $_{\text{seller}}$]
- 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

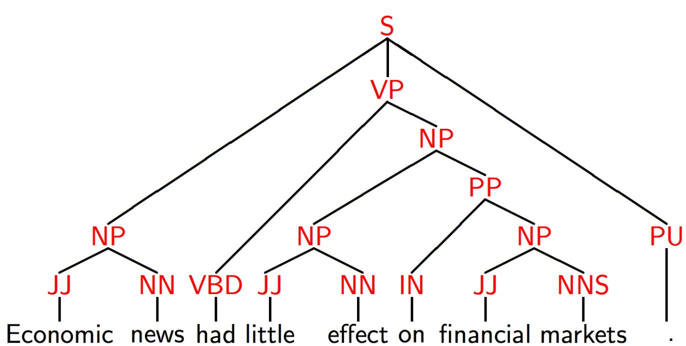
He	*goes*	*on*	*the*	*road*	*with*	*his*	*piano*
NP	$(S[dcl]\backslash NP)/PP$	PP/NP	NP/N	N	$((S\backslash NP)\backslash(S\backslash NP))/NP$	NP/N	N

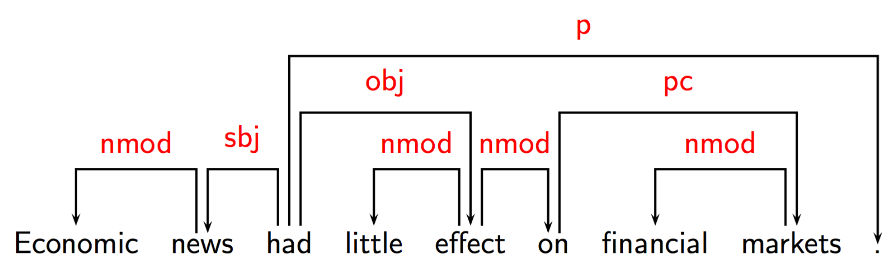
A	*bitter*	*conflict*	*with*	*global*	*implications*
NP/N	N/N	N	$(NP\backslash NP)/NP$	N/N	N

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%)

Parsing Algorithms

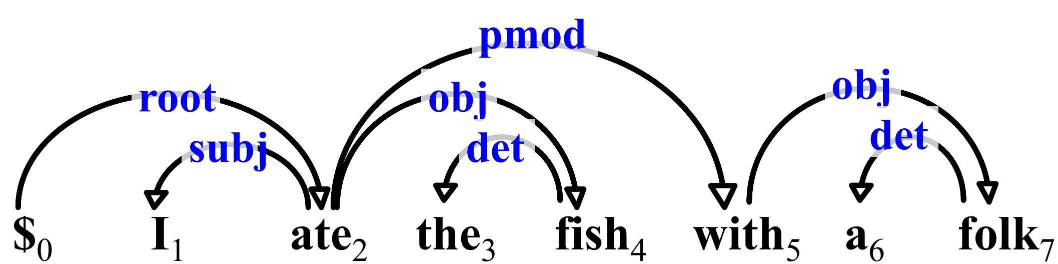
- All kinds of algorithms converting sentences to tree or graph structures
  - Constituency and Dependency Parsing





Dependency Parsing Evaluation Metrics

- Unlabeled attachment score (UAS)
  - The percent of words that have the correct heads
- Labeled attachment score (LAS)
  - The percent of words that have the correct heads and labels.
- Root Accuracy (RA)
- Complete Match rate (CM)



Conclusion

- NLP Tasks
  - Word segmentation, POS tagging, named entity recognition
  - Constituent/dependency parsing
  - Semantic Role Labeling, Semantic (graph) dependency parsing
  - AMR, CCG
- Structured Prediction
  - Sequence segmentation
  - Sequence labeling / Tagging
  - Parsing

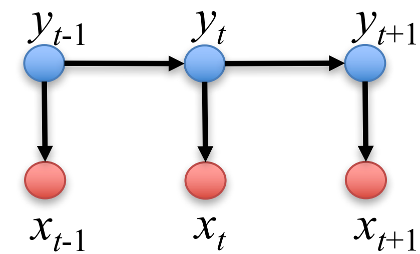
Part 2: Graph-based Methods

Part 2.1: Graph-based Sequence Labeling

Sequence Labeling Models

HMM

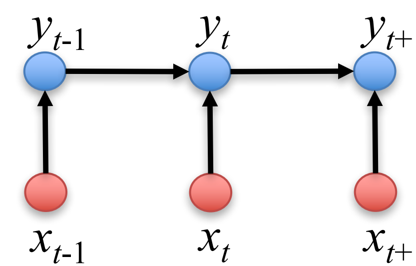
$$P(y_{[1:n]}, x_{[1:n]}) \propto \prod_{t=1}^{n} P(y_t | y_{t-1}) P(x_t | y_t)$$



MEMM

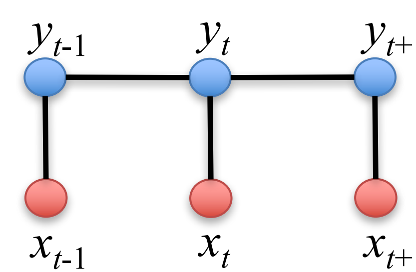
$$P(y_{[1:n]} | x_{[1:n]}) \propto \prod_{t=1}^{n} P(y_t | y_{t-1}, x_t)$$

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



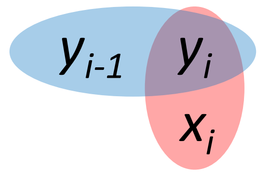
CRF

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



Features of POS Tagging with CRF

- Assume only two feature templates
  - tag bigrams
  - word/tag pairs



$$f_{100} = \begin{cases} 1 \text{ if } < y_{i-1}, y_i > \; = \; < n, v > \\ \quad\quad 0 \text{ otherwise} \end{cases}$$

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

CRF Decoding

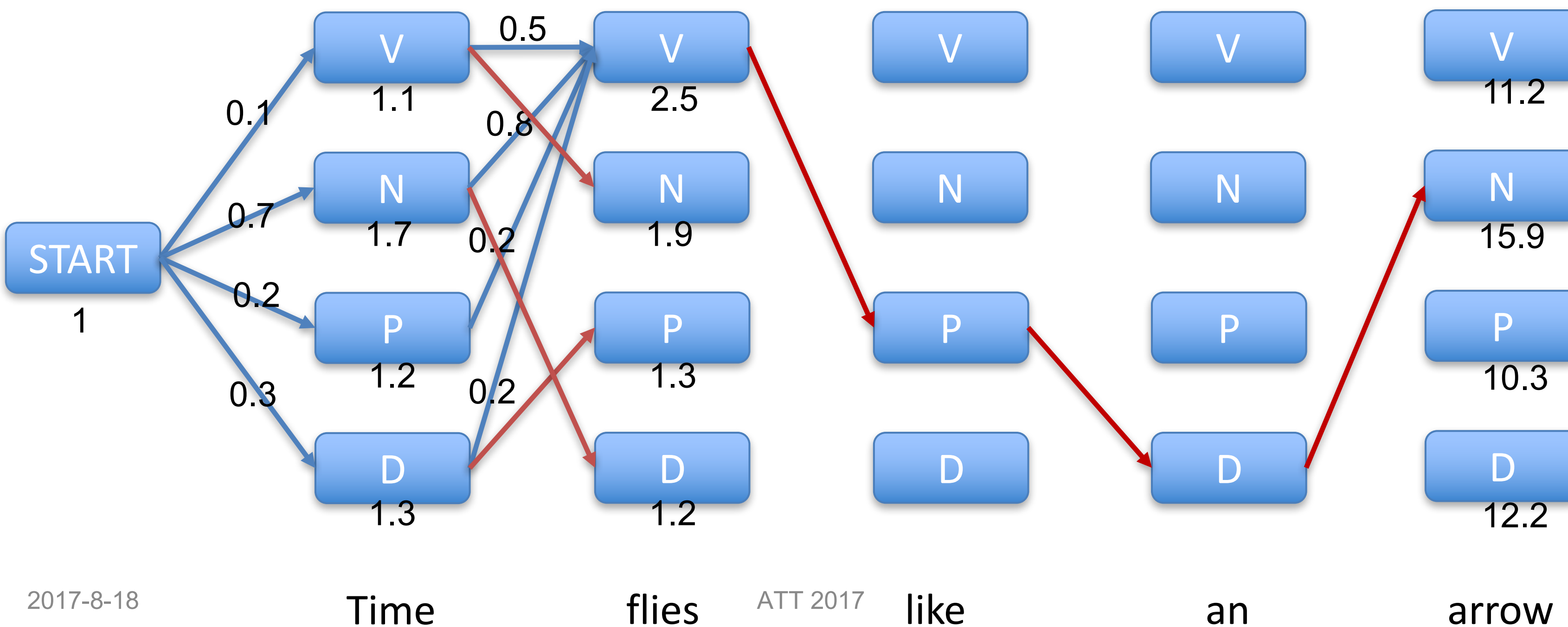
$$\underset{y_{[1:n]} \in \mathrm{GEN}(x_{[1:n]})}{\arg\max} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(x_{[1:n]}, y_i, y_{i-1})$$

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

- Dynamic Programming Algorithm
  - Viterbi Algorithm

Viterbi Algorithm

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



Constituency Parsing

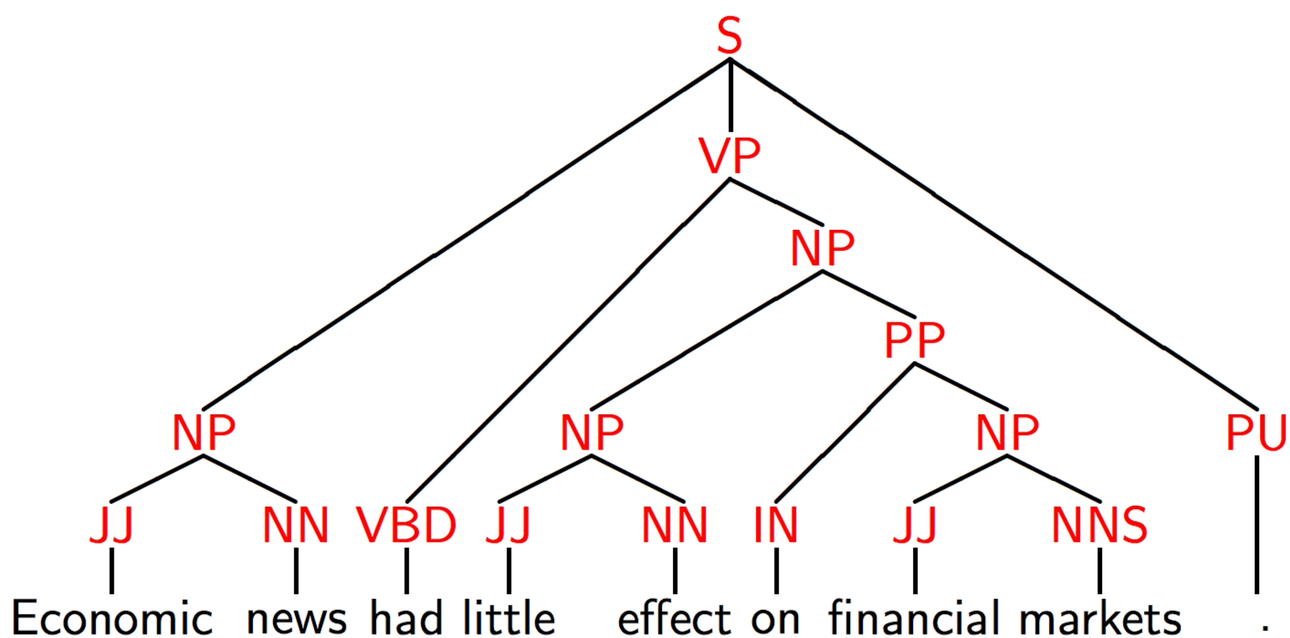
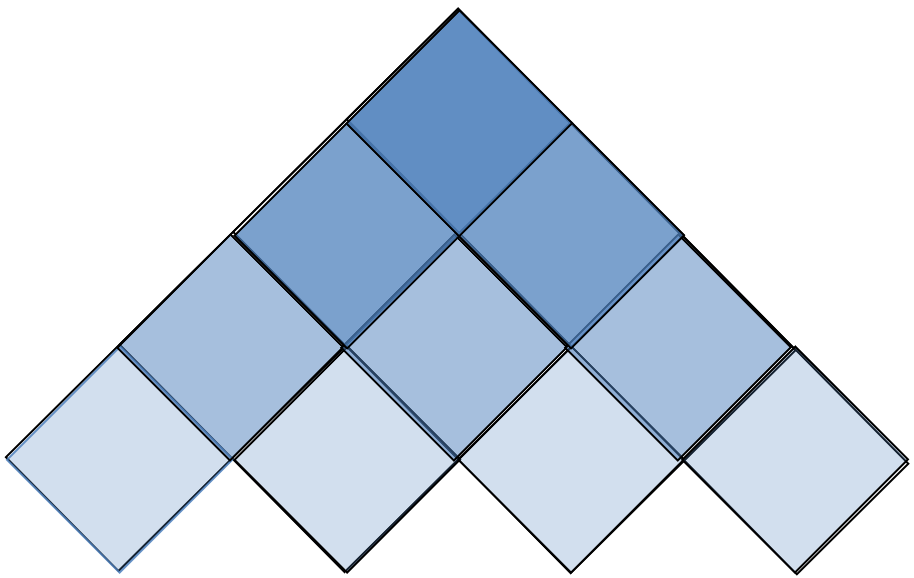


Chart-based Method

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



fish people fish tanks

PCFG

Rule	**Prob θ_i**
$S \rightarrow NP\ VP$	θ_0
$NP \rightarrow NP\ NP$	θ_1
…	
$N \rightarrow$ fish	θ_{42}
$N \rightarrow$ people	θ_{43}
$V \rightarrow$ fish	θ_{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

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

and

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

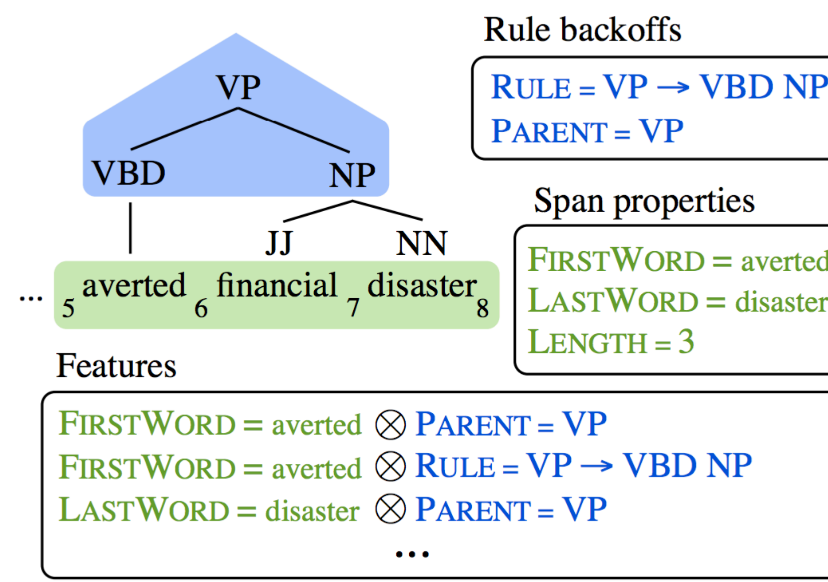
Output: Return $\pi(1, n, S) = \max_{t \in \mathcal{T}(s)} p(t)$, and backpointers bp which allow recovery of $\arg\max_{t \in \mathcal{T}(s)} p(t)$.

Constituency Parsing with CRF

- Probability of a tree *T* conditioned on a sentence **w**

$$p(T|\mathbf{w}) \propto \exp\left(\theta^{\mathsf{T}} \sum_{r \in T} f(r, \mathbf{w})\right)$$

- More features

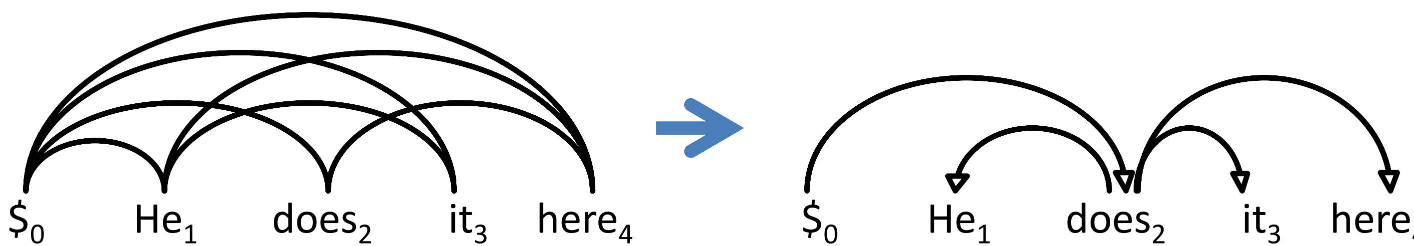


Hall D, Durrett G, Klein D (2014) Less grammar, more features. ACL.

Part 2.2: Graph-based Dependency Parsing

Graph-based Dependency Parsing

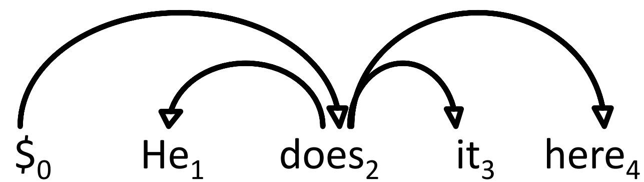
- Find the highest scoring tree from a complete dependency graph



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

First-order as an Example

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



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

How to Score an Arc

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

$$score(2,4) = ?$$

$\$_0$ He_1 $does_2$ it_3 $here_4$

- Feature based representation: an arc is represented as a feature vector **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]
[VERB, *, IN, left, 5]	[VERB, JJ, IN, left, 5]	[VERB, JJ, *, left, 5]	[NOUN, *, IN, left, 5]	[NOUN, VERB, IN, left, 5]

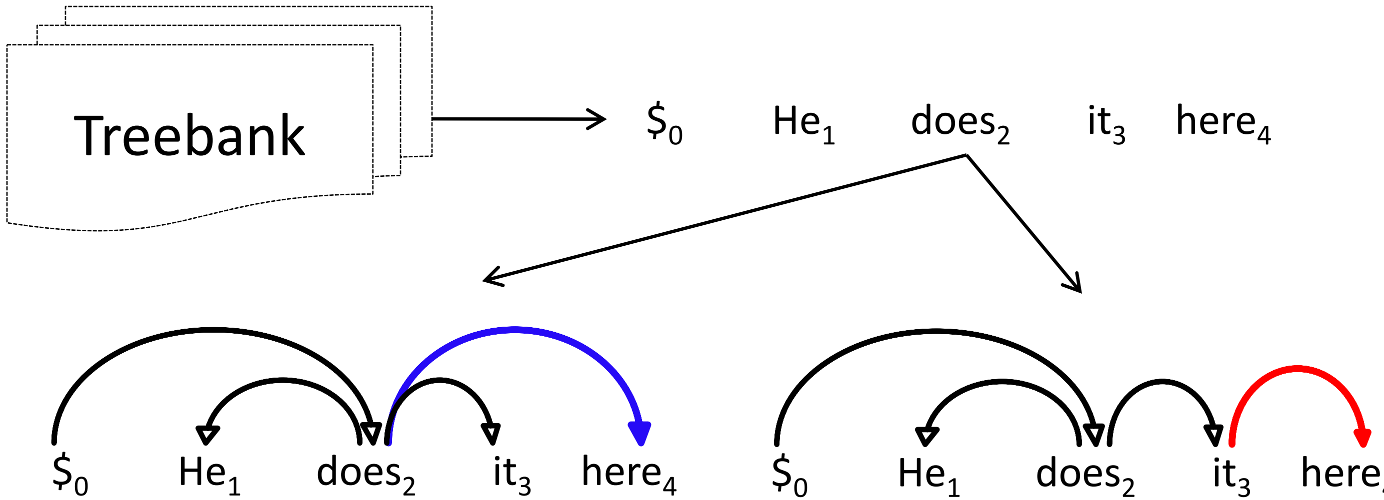
Decoding for first-order model

- Maximum Spanning Tree (MST) Algorithm
- 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

How to learn **w**?

- Use a treebank
  - Each sentence has a manually annotated dependency tree.
- Online training (Collins, 2002; Crammer and Singer, 2001; Crammer+, 2003)
  - Initialize w = 0
  - Go though the treebank for a few (10) iterations.
    - Use one instance to update the weight vector.

Online learning **w**



Gold-standard parse Y^+

1-best parse Y^- with $\mathbf{w}^{(k)}$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(X,Y^+) - \mathbf{f}(X,Y^-)$$

Conclusion

- Graph-based Sequence Labeling
  - HMM → MEMM → CRF
- Graph-based Dependency Parsing
  - Scoring function
  - DP Decoding
  - Online learning

Part 3: Transition-base Methods

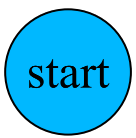
Part 3.1: Transition Systems

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

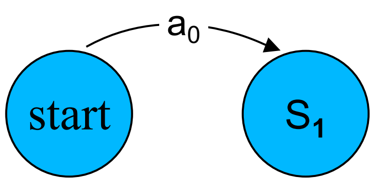
A transition system

- Automata



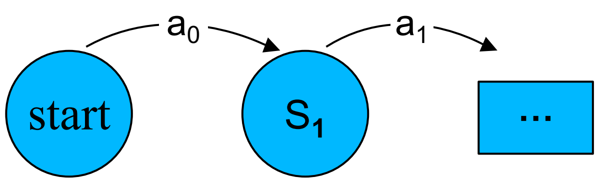
A transition system

- Automata



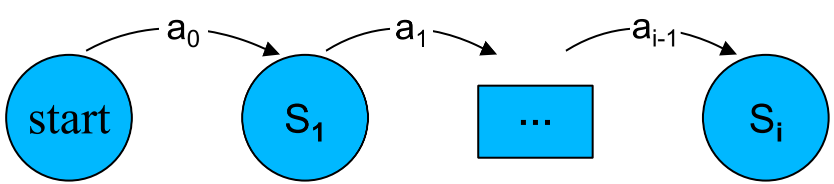
A transition system

- Automata



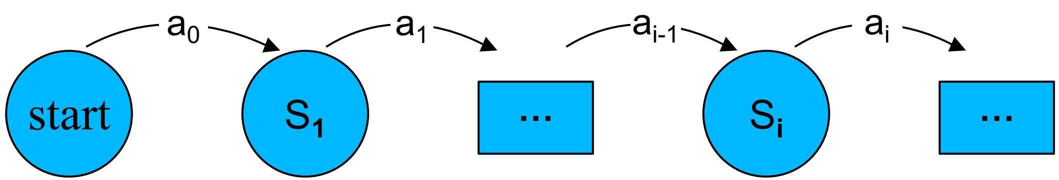
A transition system

- Automata



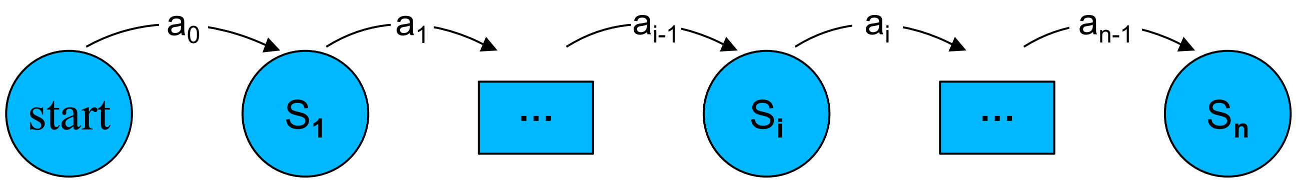
A transition system

- Automata



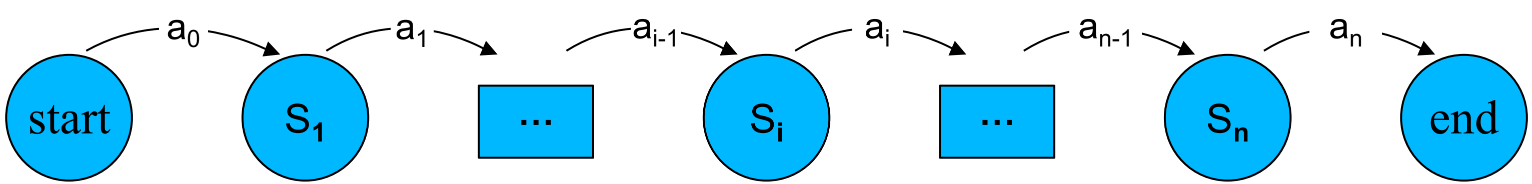
A transition system

- Automata



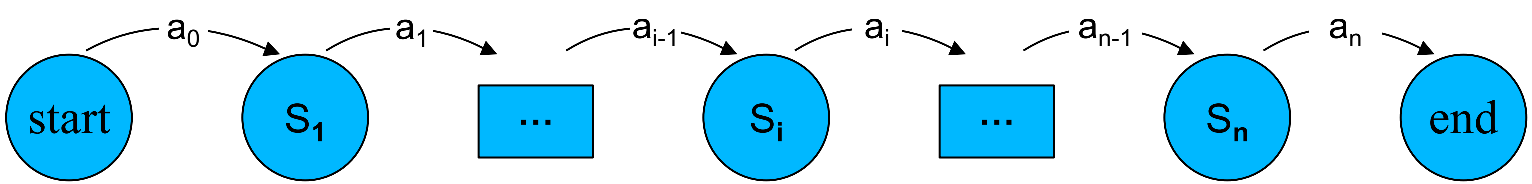
A transition system

- 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

Part 3.2: Transition-base Dependency Parsing

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

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

$a_i \rightarrow$ the action adopted in step i

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

$Y \rightarrow$ the tree built by the action sequence $a_0 \ldots a_m$

Transition-based Dependency Parsing

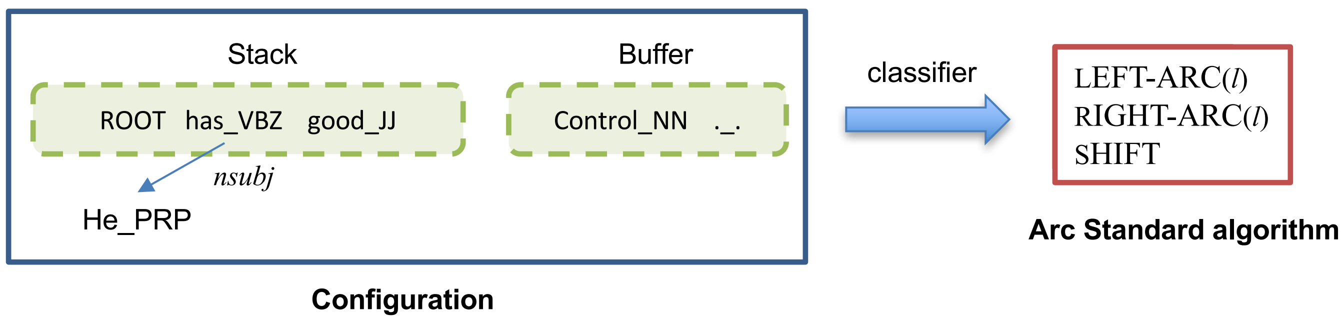
- The goal of a transition-based dependency parser is to find the highest scoring action sequence that builds a legal tree.

$$
\begin{aligned}
Y^* &= \mathop{\arg\max}_{Y \in \Phi(X)} score(X, Y) \\
&= \mathop{\arg\max}_{a_0 \ldots a_m \to Y} \sum_{i=0}^{m} score(X, h_i, a_i)
\end{aligned}
$$

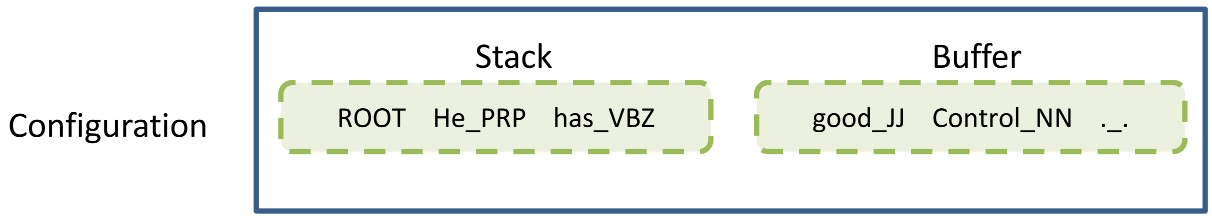
Transition-based Dependency Parsing

- Greedily predict a transition sequence from an initial parser state to some terminal states
- State (configuration)

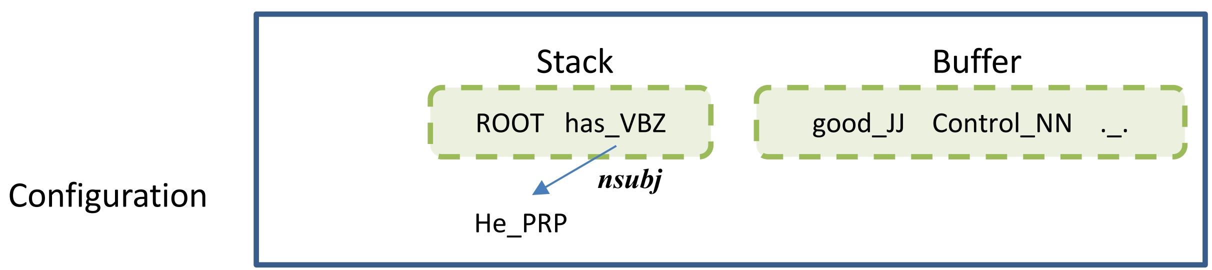
  = Stack + Buffer + Dependency Arcs



Transition Action: LEFT-ARC (*l*)

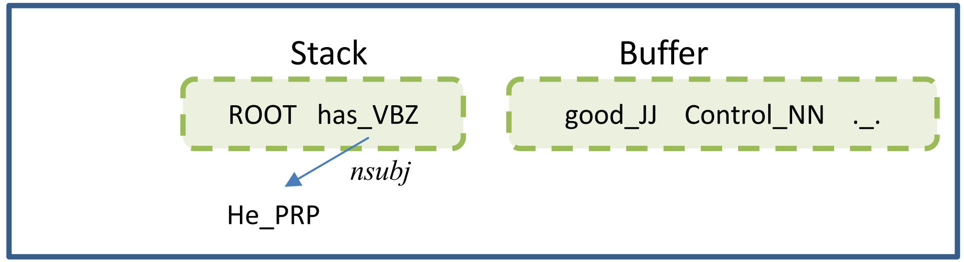


Operation:
- Add a left arc (S_0)
- Remove "He_PRP" from Stack



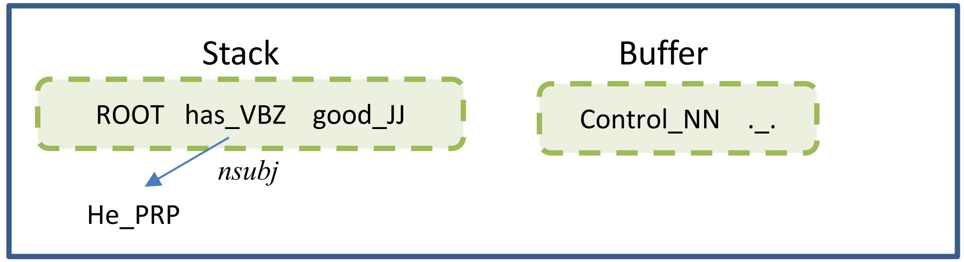
Transition Action: SHIFT

Configuration



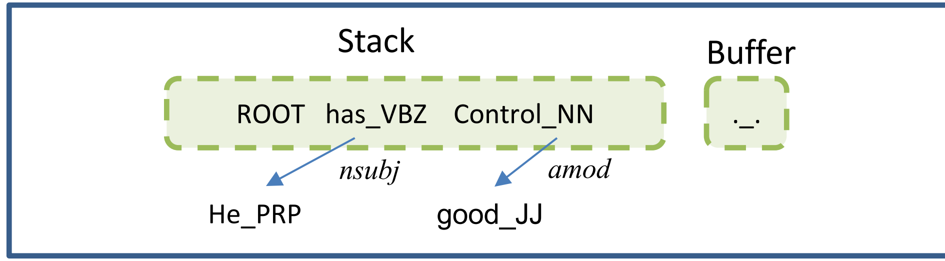
Operation:
- Shift "good_JJ" from Buffer to top of Stack

Configuration



Transition Action: RIGHT-ARC (I)

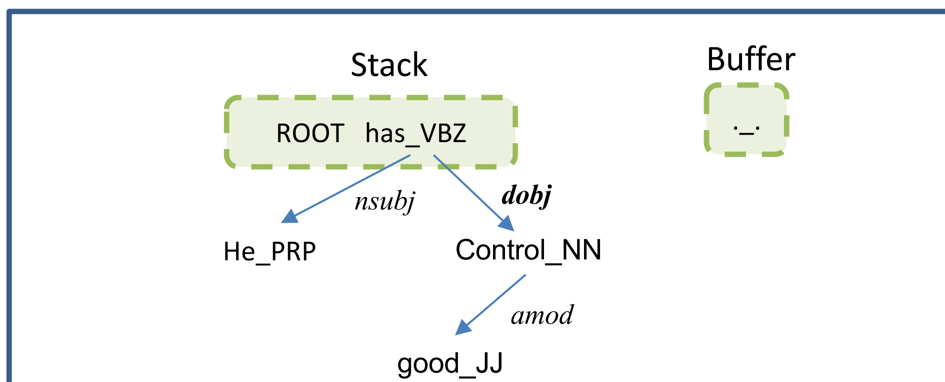
Configuration



Operation:

- Add a right arc (S_1)
- Remove S_0 (“Control_NN”) from Stack

Configuration



An Example

Arc-standard Algrithm

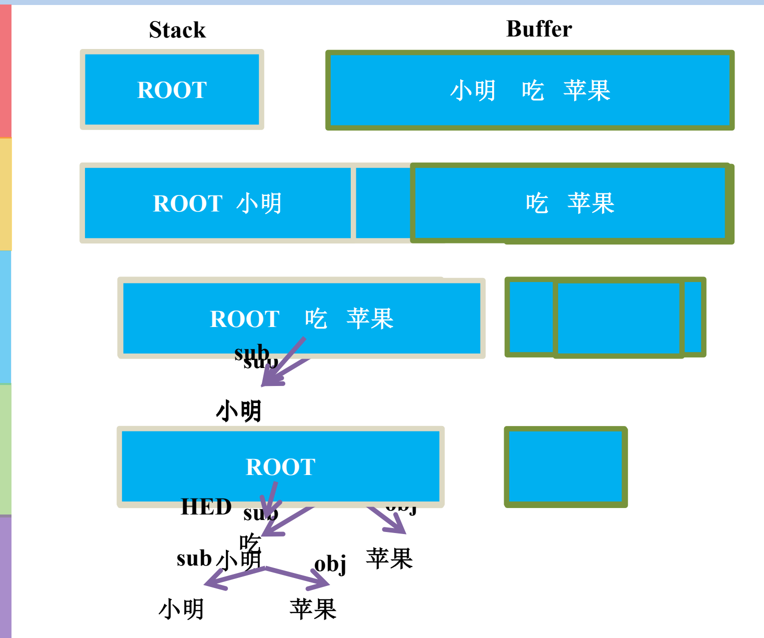
初始状态
Stack只有根节点，待处理词在Buffer中

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

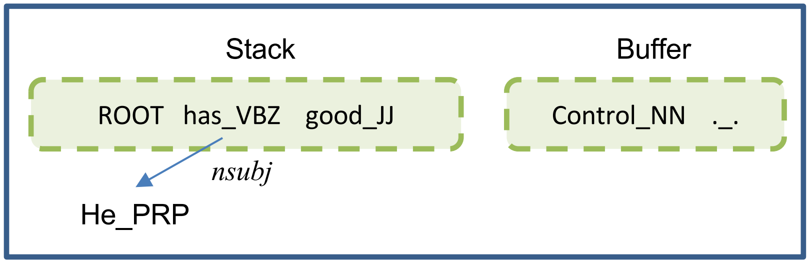
RIGHT-ARC
弹出栈顶词，生成一条弧从栈顶第二个词指向栈顶词

终结状态
Stack只有根节点，Buffer为空



Traditional Features

Configuration



Feature Vector:
- Binary
- Sparse
- High-dimensional

0	0	1	0	1	...	0	1	0	0

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
$N_0pN_1pN_2p$; $S_0pN_0pN_1p$; $S_{0h}pS_0pN_0p$; $S_0pS_{0l}pN_0p$; $S_0pS_{0r}pN_0p$; $S_0pN_0pN_{0l}p$

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
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$;$N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
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.

Part 3.3: Transition-base Methods for More Tasks

A transition-based POS-tagging example

- POS tagging

  I like reading books $\rightarrow$ I/PRON like/VERB reading/VERB books/NOUN

- Transition system
  - State
    - Partially labeled word-POS pairs
    - Unprocessed words
  - Actions
    - TAG(t) $w_1/t_1 \cdots wi/t_i \rightarrow w_1/t_1 \cdots w_i/t_i\ w_{i+1}/t$

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

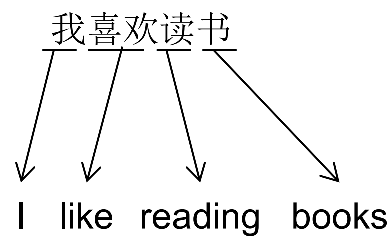
- State
  - Partially segmented results
  - Unprocessed characters

- Two candidate actions
  - Separate ## ## → ## ## #
  - Append ## ## → ## ##‍#

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- Initial State



Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- Separate

我

喜欢读书

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- Separate

我　喜　　　　欢读书

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- Append

我　喜欢　　　　　　　　读书

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- Separate

我　喜欢　读　　　　书

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- Separate

我 喜欢 读 书

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

Word segmentation

- End State

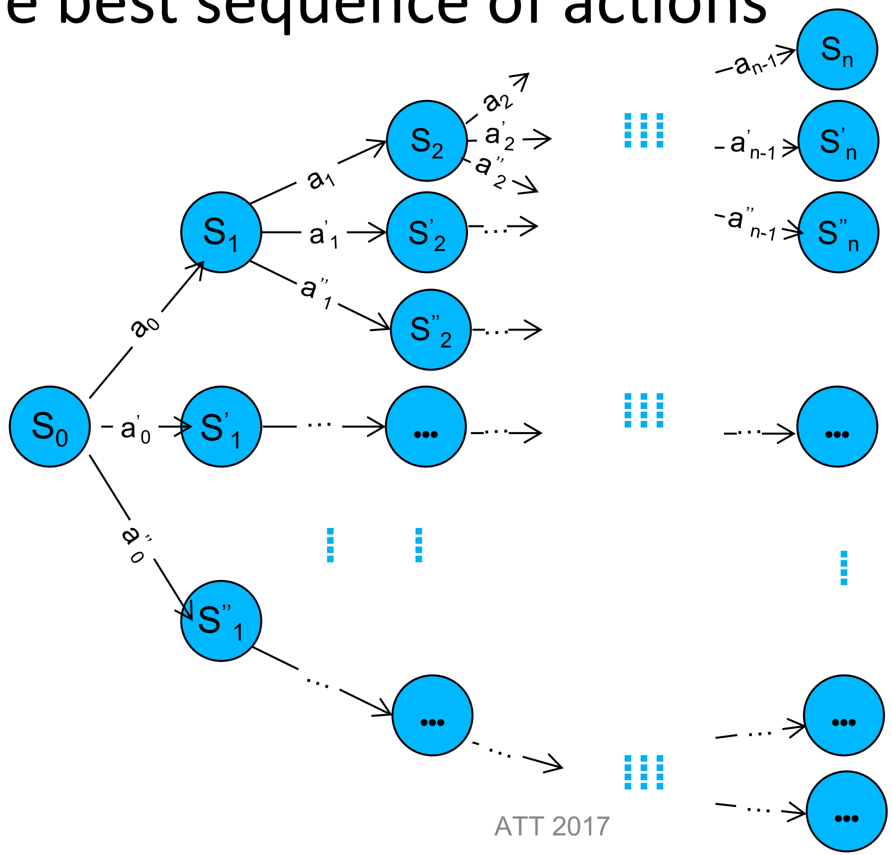
我　喜欢　读　书

Zhang, Y., & Clark, S. (2007). Chinese Segmentation Using a Word-Based Perceptron Algorithm. ACL.

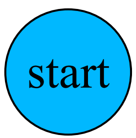
Part 3.4: Transition-base Methods with Beam-search Decoding

Search

- Find the best sequence of actions

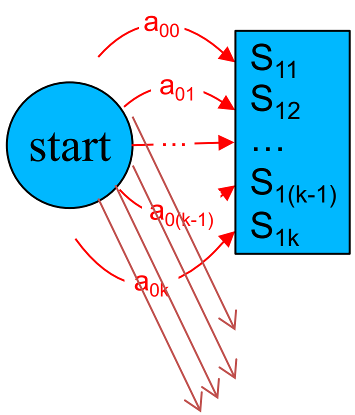


Beam-search decoding



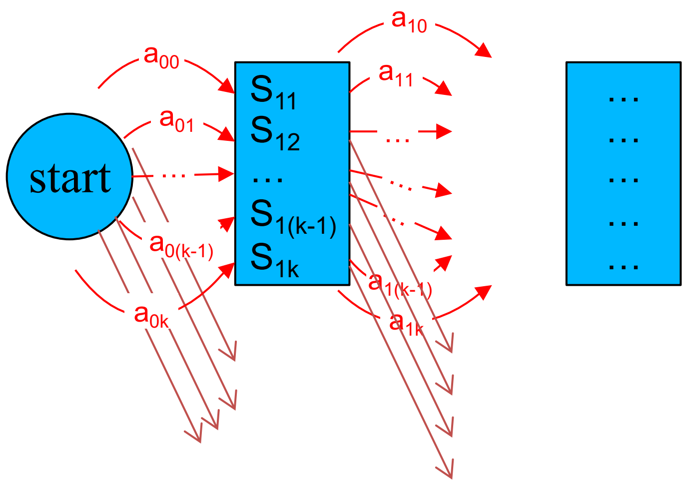
Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Beam-search decoding



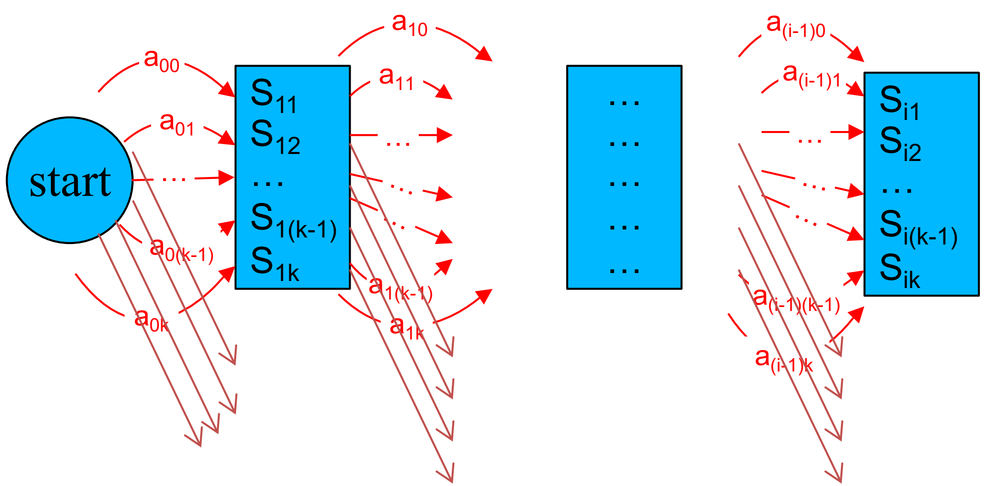
Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Beam-search decoding



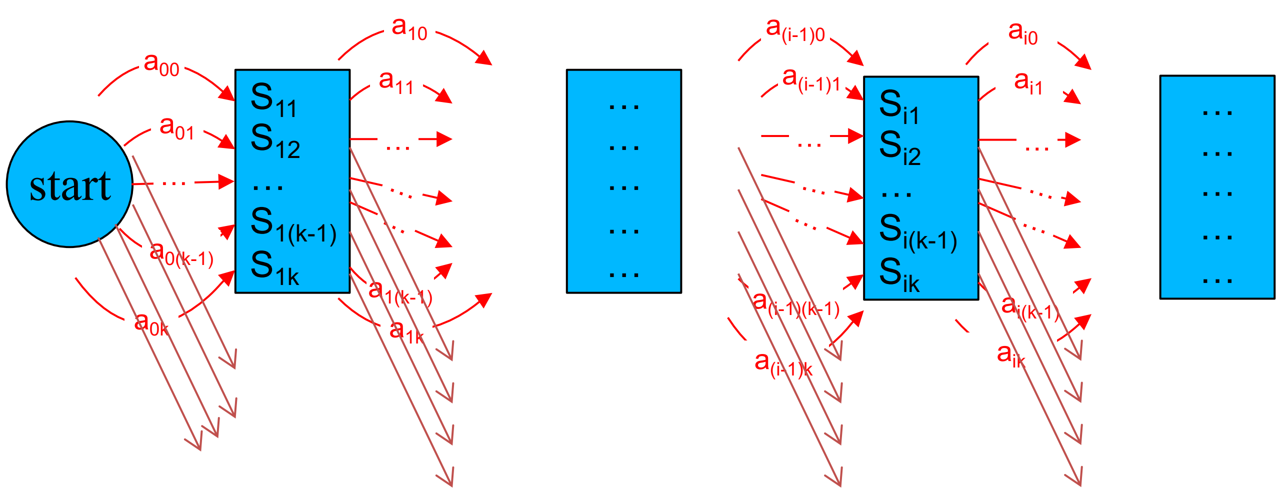
Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Beam-search decoding



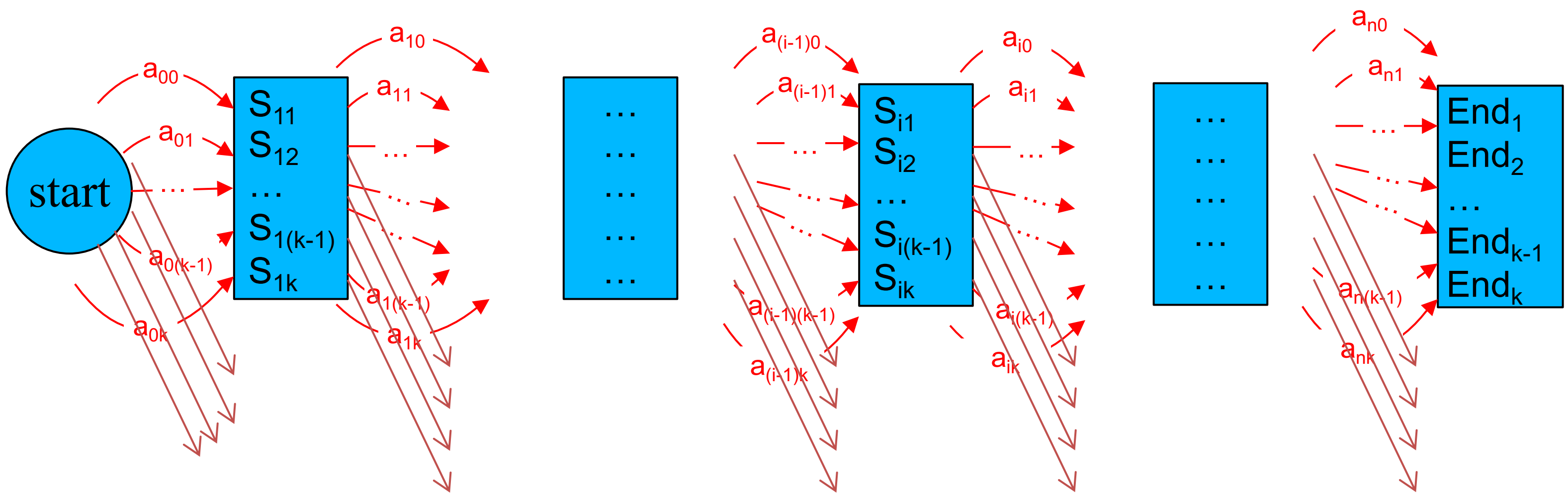
Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Beam-search decoding



Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Beam-search decoding



Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Beam-search decoding

function BEAM-SEARCH(*problem*, *agenda*, *candidates*, *B*)

  candidates ← {STARTITEM(*problem*)}
  agenda ← CLEAR(*agenda*)
  loop do
    for each *candidate* **in** *candidates*
      agenda ← INSERT(EXPAND(*candidate*, *problem*), *agenda*)
    best ← TOP(*agenda*)
    if GOALTEST(*problem*, *best*)
      then return *best*
    candidates ← TOP-B(*agenda*, *B*)
    agenda ← CLEAR(*agenda*)

Zhang, Y., & Clark, S. (2011). Syntactic Processing Using the Generalized Perceptron and Beam Search. CL.

Conclusion

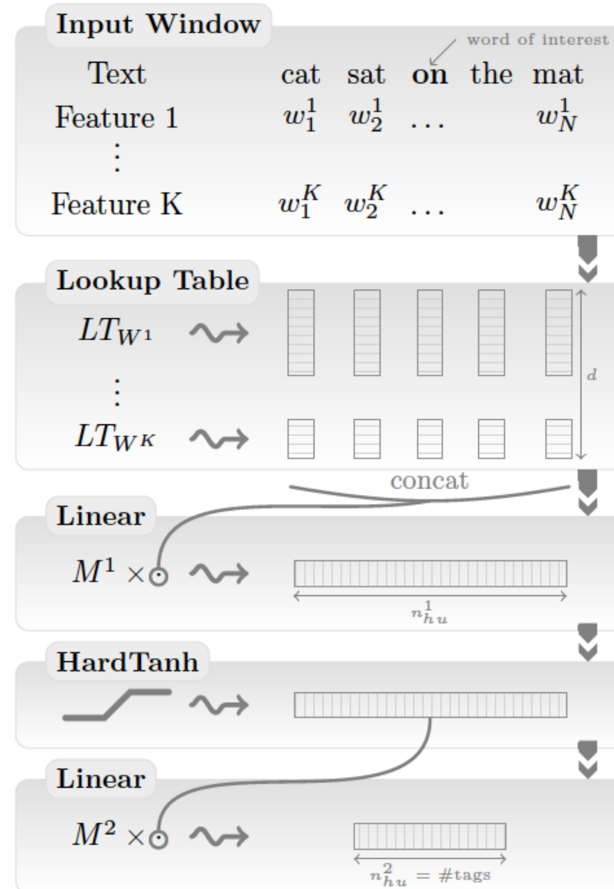
- Transition-based Dependency Parsing
  - Transition system
  - Features
- Transition-based POS Tagging and Word Segmentation
- Beam-search Decoding

Part 4: Neural Graph-based Methods

Part 4.1: Neural CRF

Window Approach for Tagging

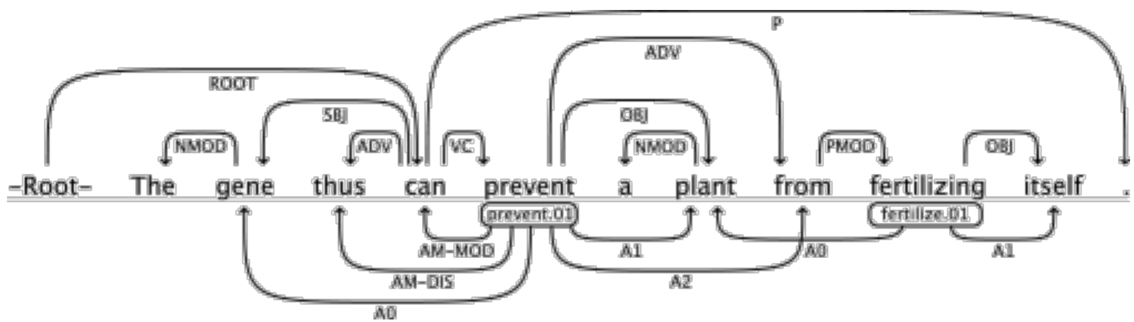
- Tasks
  - POS tagging, Chunking, NER, SRL
- Tag **one word** at a time
- Feed a **fixed-size** window of text around
- Features
  - Words, POS tags, Suffix, Cascading, …



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 for Tagging

- 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!



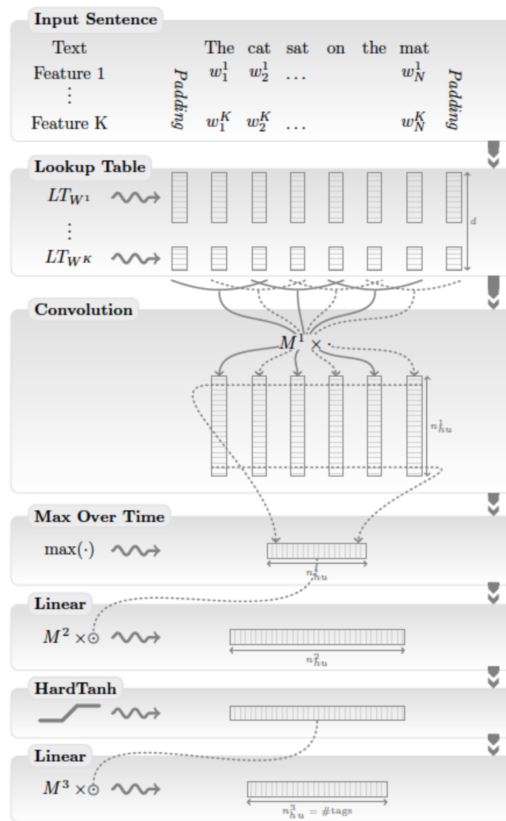
Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. J. Mach. Learn. Res. 12, 2493-2537.

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

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.

Sentence Approach



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.

Results

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
Benchmark Systems	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

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.

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

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

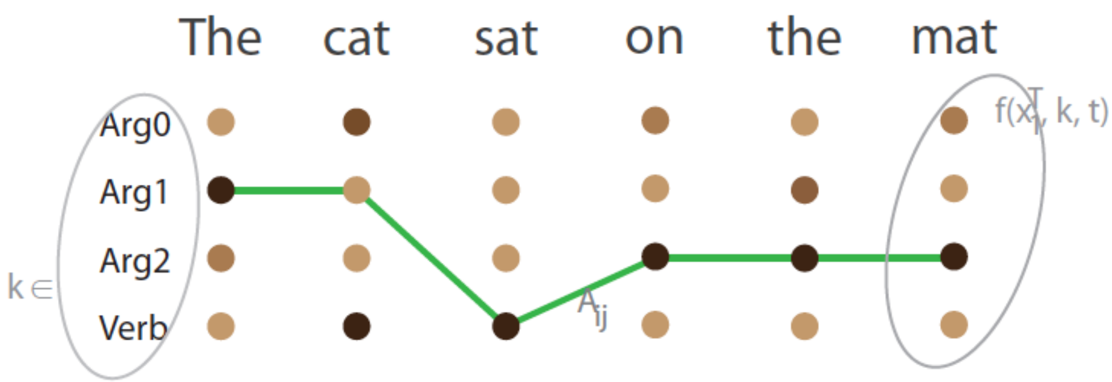
$$s([\boldsymbol{x}]_1^T, [i]_1^T, \tilde{\boldsymbol{\theta}}) = \sum_{t=1}^{T} \Big(A_{[i]_{t-1}[i]_t} + f([\boldsymbol{x}]_1^T, [i]_t, t, \boldsymbol{\theta}) \Big)$$

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

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.

Sentence-Level Log-Likelihood

- Decoding: finding the max scored path
  - Viterbi algorithm



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.

Results

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
Benchmark Systems	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

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.

Improvements

- Supervised word embeddings

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	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
Benchmark Systems	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
SENNA	32	4

(a) POS

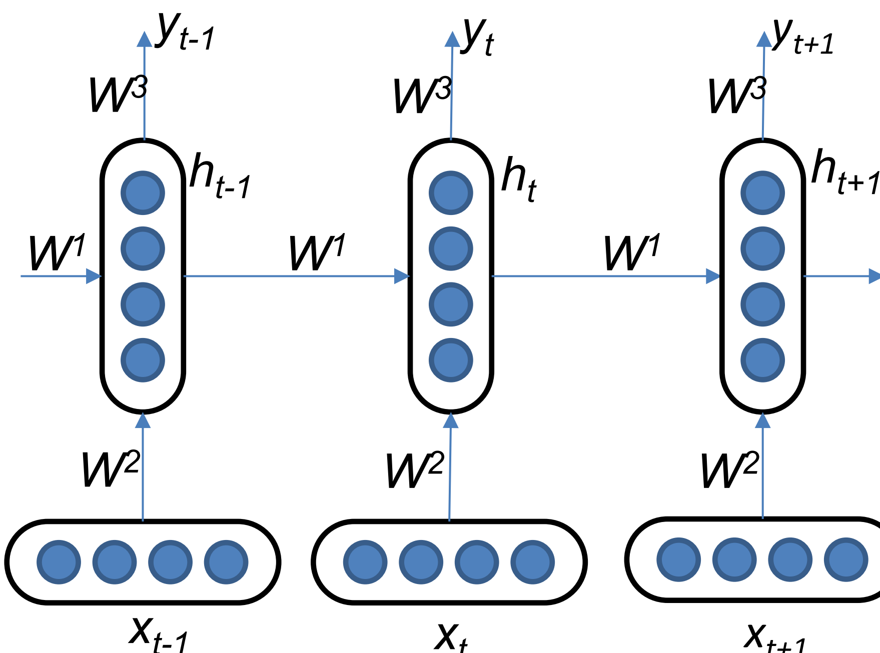
System	RAM (Mb)	Time (s)
Koomen, 2005	3400	6253
SENNA	124	52

(b) SRL

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.

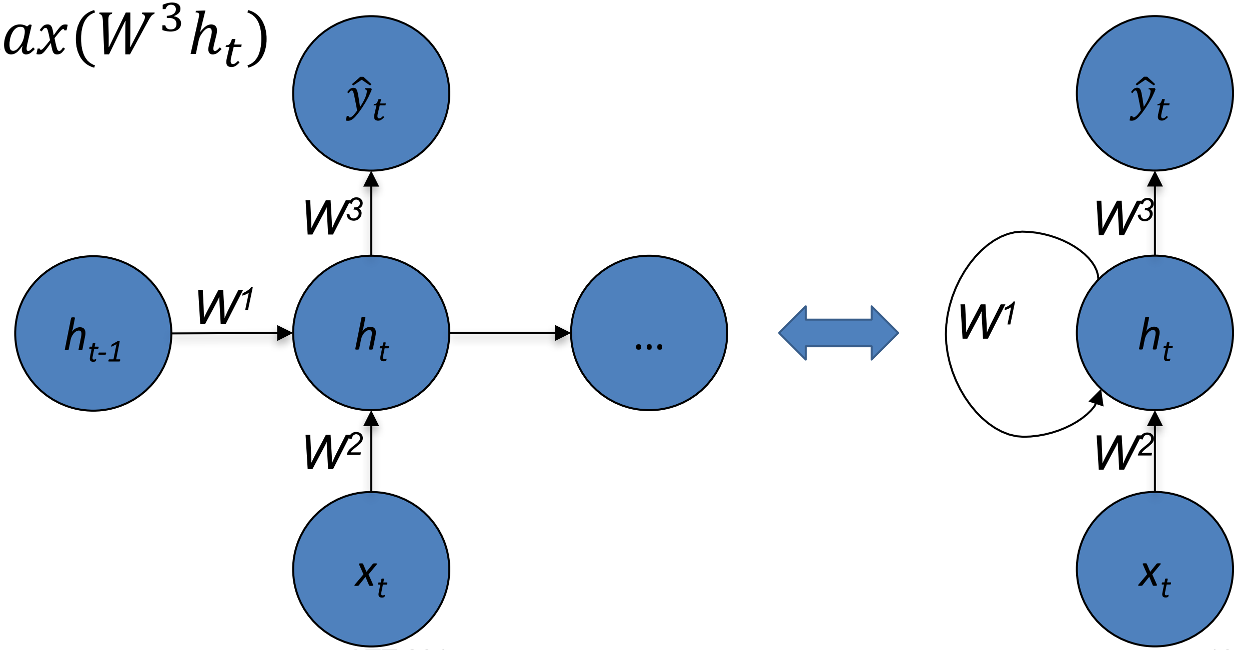
Recurrent Neural Networks (RNNs)

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



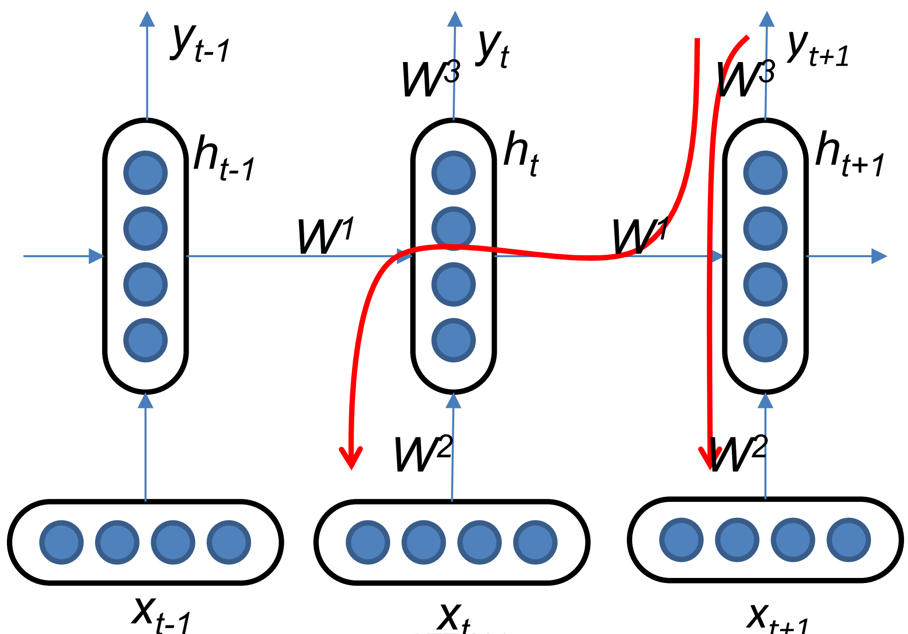
Recurrent Neural Networks (RNNs)

- At a single time step t
  - $h_t = \tanh(W^1 h_{t-1} + W^2 x_t)$
  - $\hat{y}_t = softmax(W^3 h_t)$

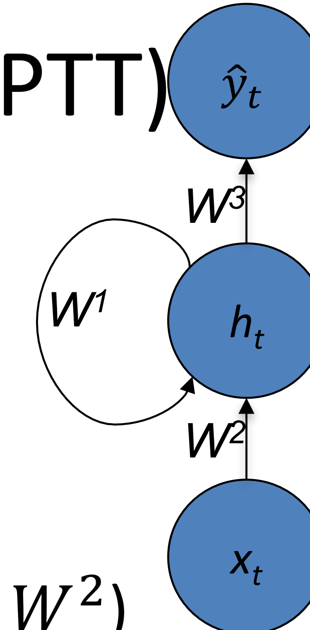


Training RNNs is hard

- 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 $\frac{\partial E_t}{\partial W^1} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W^1}$

BackPropagation Through Time (BPTT)

- Use the same as the backpropagation algorithm as we use in deep feedforward NN, but summing up the gradients for W^1
- BPTT is just a fancy name for standard backpropagation on an unrolled RNN
- $\frac{\partial E}{\partial W^1} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W^1}$
- $\frac{\partial E_t}{\partial W^1} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W^1}$

The vanishing gradient problem

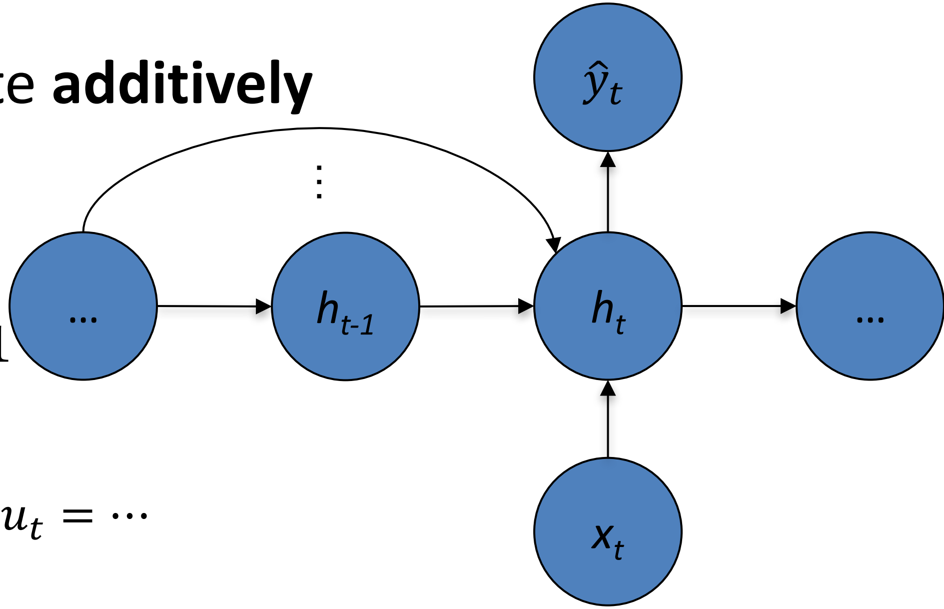
- $\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W}$, $h_t = \tanh(W^1 h_{t-1} + W^2 x_t)$
- $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^1 \mathrm{diag}[\tanh'(\cdots)]$
- $\left\| \frac{\partial h_t}{\partial h_{t-1}} \right\| \leq \gamma \|W^1\| \leq \gamma \lambda_1$
  - where γ is bound $\|\mathrm{diag}[\tanh'(\cdots)]\|$, λ_1 is the largest singular value of W^1
- $\left\| \frac{\partial h_t}{\partial h_k} \right\| \leq (\gamma \lambda_1)^{t-k}$→0, if $\lambda_1 < \frac{1}{\gamma}$
- This can become very small or very large quickly → Vanishing or exploding gradient
  - Trick for exploding gradient: clipping trick (set a threshold)

A “solution”

- Intuition
  - Ensure $\gamma\lambda_1 \geq 1$ → 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
    - $h_t = h_{t-1} + u_t = h_{t-2} + u_{t-1} + u_t = \cdots$



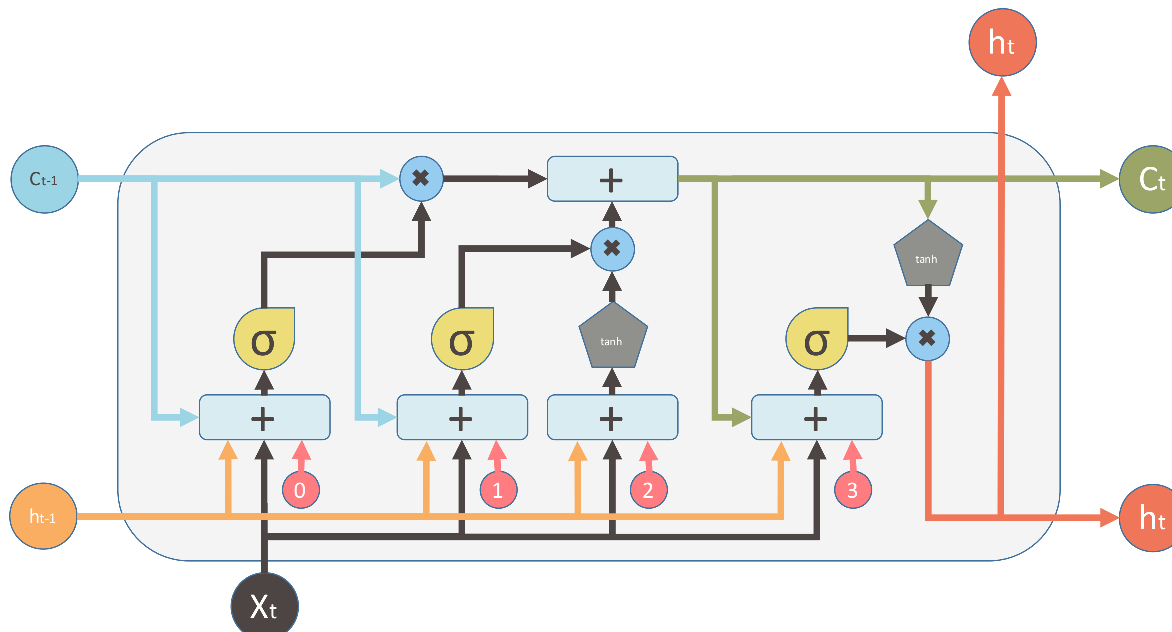
A better "solution" (cont.)

- Interpolate between old state and new state ("choosing to **forget**")
  - $f_t = \sigma\left(W^f x_t + U^f h_{t-1}\right)$
  - $h_t = f_t \odot h_{t-1} + (1 - f_t) \odot u_t$
- Introduce a separate **input gate** i_t
  - $i_t = \sigma\left(W^i x_t + U^i h_{t-1}\right)$
  - $h_t = f_t \odot h_{t-1} + i_t \odot u_t$
- 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)

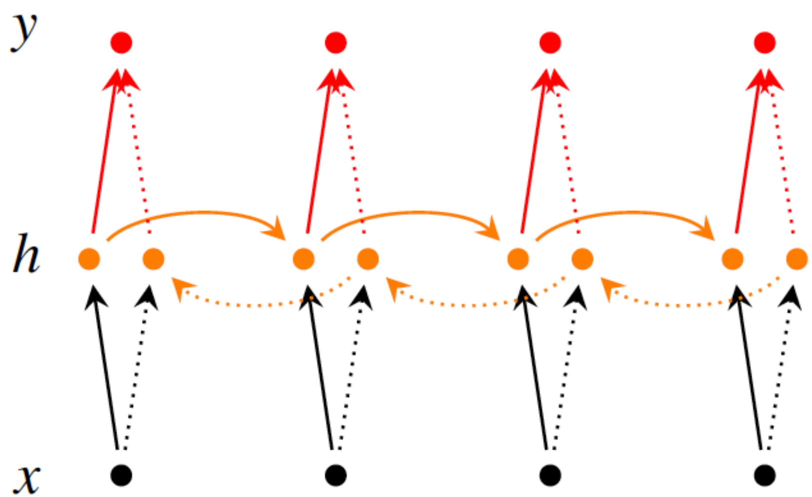
- Hochreiter & Schmidhuber, 1997
- LSTM = additive updates + gating

$$u_t = \tanh\left(W h_{t-1} + V x_t\right)$$
$$f_t = \text{sigmoid}\left(W_f h_{t-1} + V_f x_t\right)$$
$$i_t = \text{sigmoid}\left(W_i h_{t-1} + V_i x_t\right)$$
$$o_t = \text{sigmoid}\left(W_o h_{t-1} + V_o x_t\right)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$
$$h_t = o_t \odot \tanh(c_t)$$
$$y_t = U h_t$$

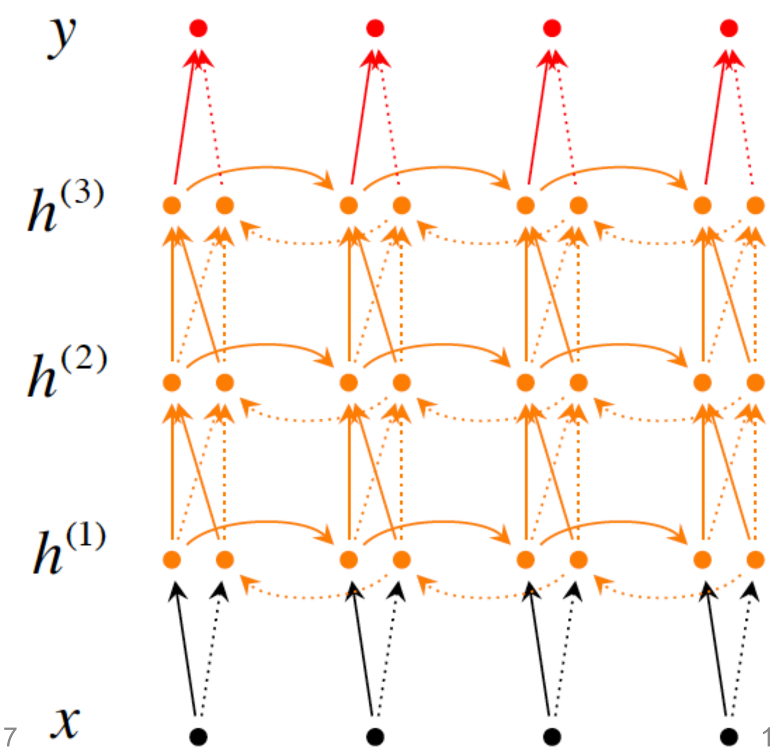


More RNNs

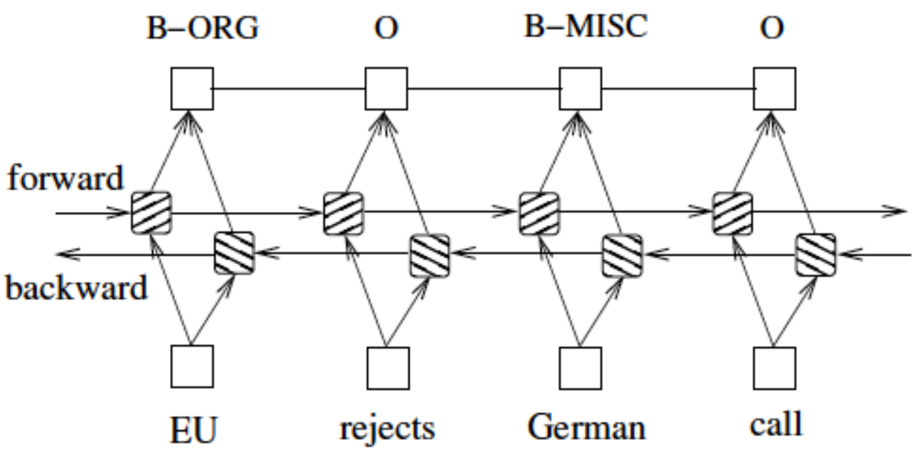
- Bidirectional RNN



- Deep Bidirectional RNN



Bi-LSTM-CRF



Algorithm 1 Bidirectional LSTM CRF model training procedure

```
for each epoch do
    for each batch do
        1) bidirectional LSTM-CRF model forward pass:
            forward pass for forward state LSTM
            forward pass for backward state LSTM
        2) CRF layer forward and backward pass
        3) bidirectional LSTM-CRF model backward pass:

            backward pass for forward state LSTM
            backward pass for backward state LSTM
        4) update parameters
    end for
end for
```

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991, 2015.

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	**97.45**	93.80	84.10
	BI-LSTM-CRF	97.43	**94.13**	**84.26**
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	**97.55**	**94.46**	**88.83 (90.10)**

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991, 2015.

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

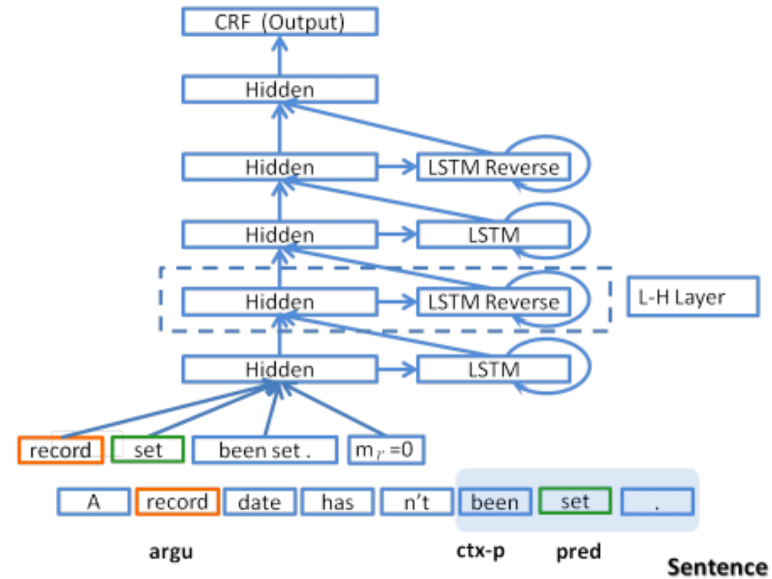
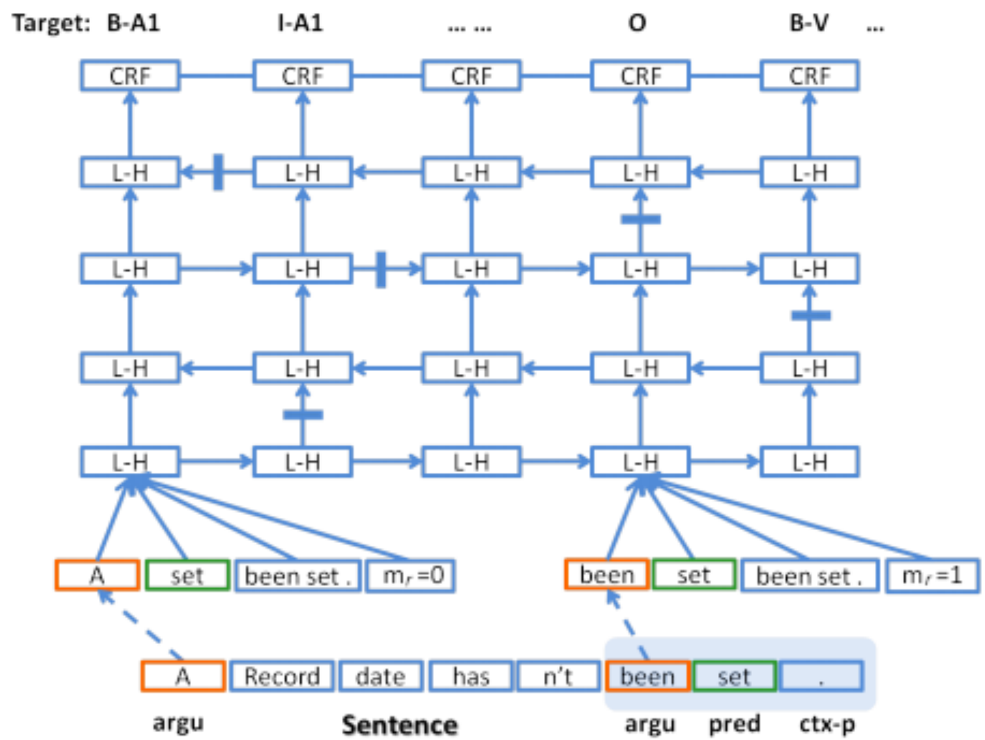


Figure 2: DB-LSTM network.Shadow part denote the predicate context within length 1.

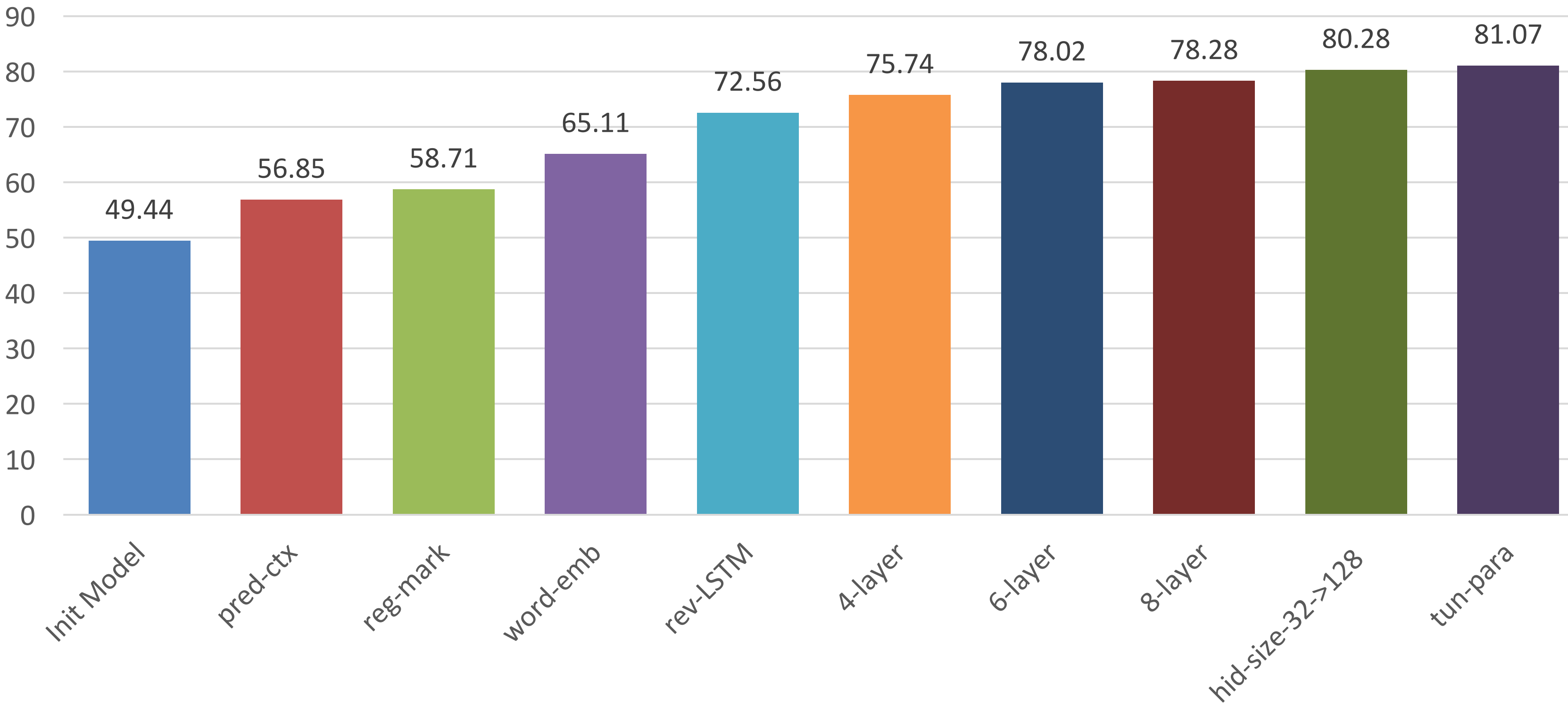
Jie Zhou and Wei Xu. (2015). End-to-end learning of semantic role labeling using recurrent neural networks. ACL.

Temporal Expanded



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

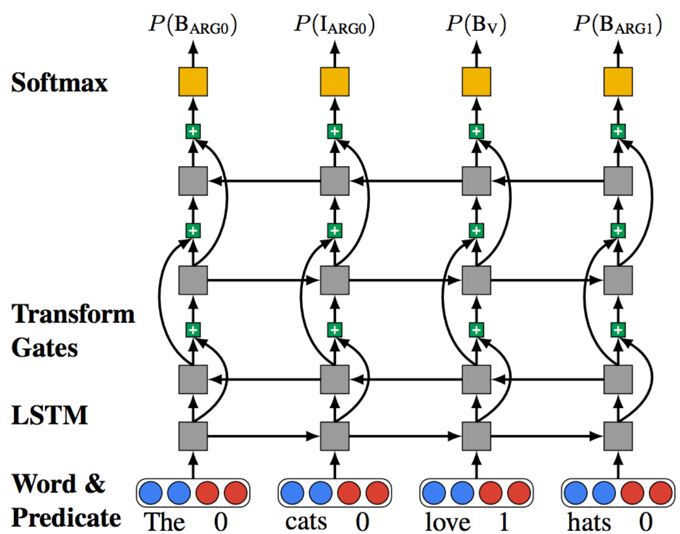
Results



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

Deep SRL

- A deep **highway** BiLSTM architecture with constraints
  - 8 BiLSTM layers (4 forward LSTMs and 4 reversed LSTMs)



Luheng He, Kenton Lee, Mike Lewis and Luke Zettlemoyer. Deep Semantic Role Labeling: What Works and What's Next. ACL 2017.

Results

- New state-of-the-art results

	Development				WSJ Test				Brown Test				Combined
Method	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.	F1
Ours (PoE)	**83.1**	**82.4**	**82.7**	**64.1**	**85.0**	**84.3**	**84.6**	**66.5**	**74.9**	**72.4**	**73.6**	**46.5**	**83.2**
Ours	81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.9	71.4	72.1	44.8	81.6
Zhou	79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-	81.1
FitzGerald (Struct.,PoE)	81.2	76.7	78.9	55.1	82.5	78.2	80.3	57.3	74.5	70.0	72.2	41.3	-
Täckström (Struct.)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8	-
Toutanova (Ensemble)	-	-	78.6	58.7	81.9	78.8	80.3	60.1	-	-	68.8	40.8	-
Punyakanok (Ensemble)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3	77.9

Luheng He, Kenton Lee, Mike Lewis and Luke Zettlemoyer. Deep Semantic Role Labeling: What Works and What's Next. ACL 2017.

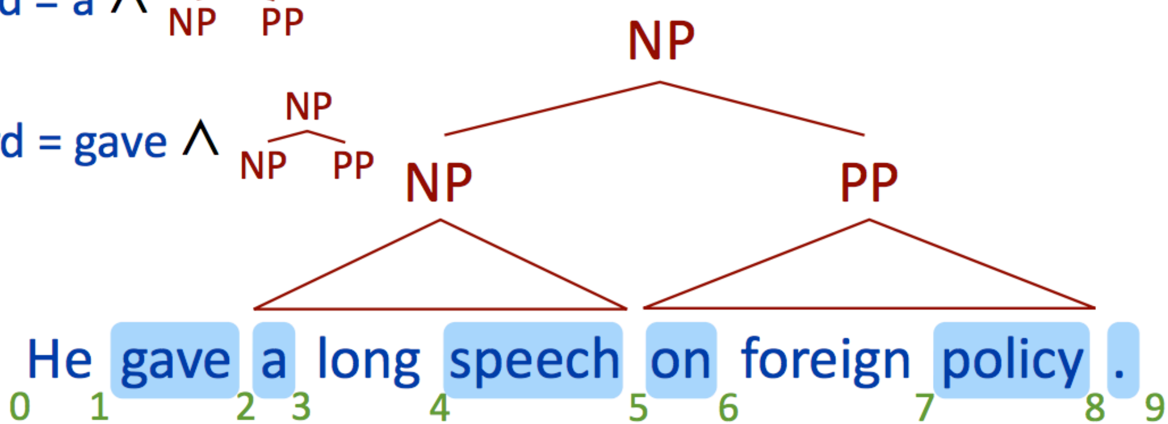
Neural CRF for Constituency Parsing

- CRF Parsing with CKY decoding

$$P(T|x) \propto \prod_{r \in T} \exp(\text{score}(r)) \qquad \text{score}\left(\begin{array}{c} \text{NP} \\ {}_2\,\text{NP}\,{}_5\,\text{PP}\,{}_8 \end{array}\right) = w^\top f\left(\begin{array}{c} \text{NP} \\ {}_2\,\text{NP}\,{}_5\,\text{PP}\,{}_8 \end{array}\right)$$

$\text{FirstWord} = \text{a} \wedge \begin{array}{c} \text{NP} \\ \text{NP} \quad \text{PP} \end{array}$

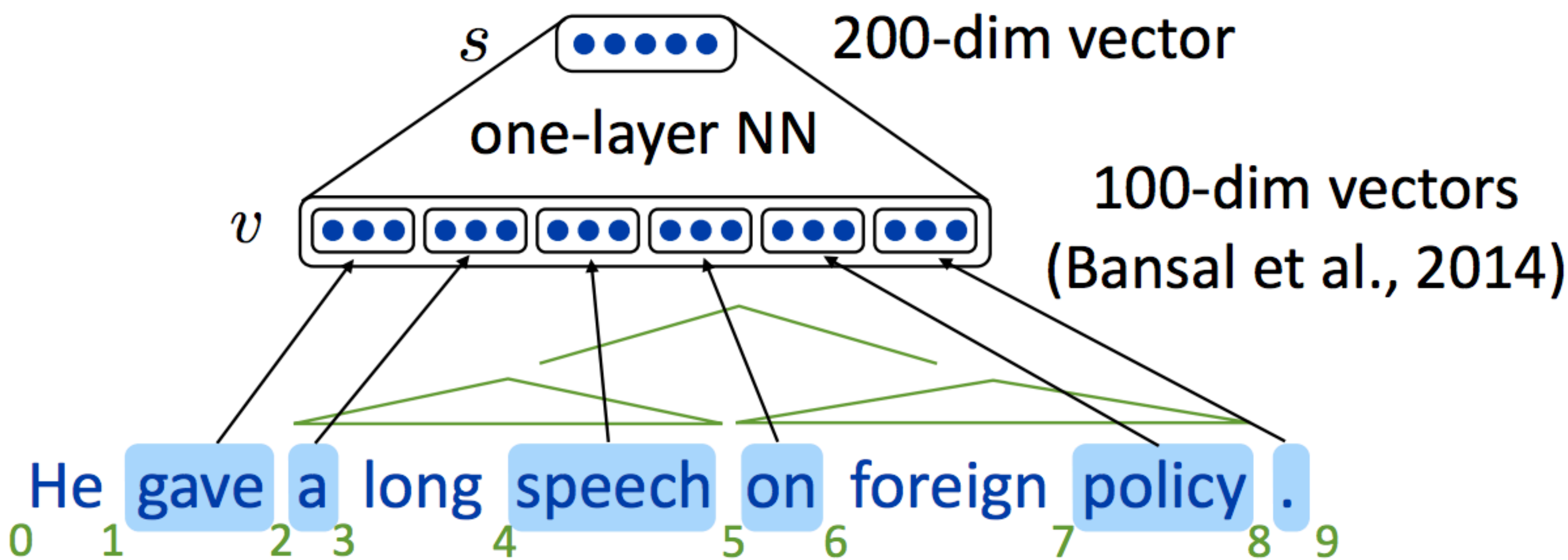
$\text{PrevWord} = \text{gave} \wedge \begin{array}{c} \text{NP} \\ \text{NP} \quad \text{PP} \end{array}$



Durrett, G., & Klein, D. (2015). Neural CRF Parsing. ACL.

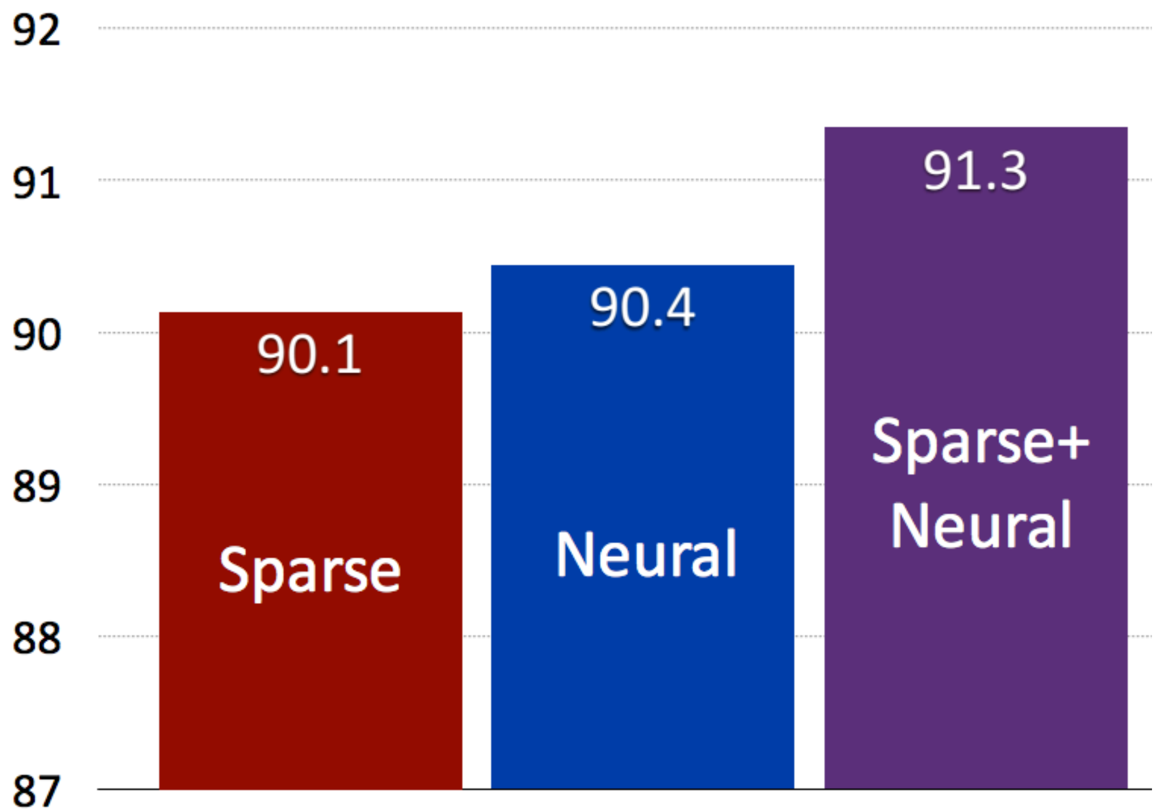
Neural CRF for Constituency Parsing

- Neural CRF Parsing



Durrett, G., & Klein, D. (2015). Neural CRF Parsing. ACL.

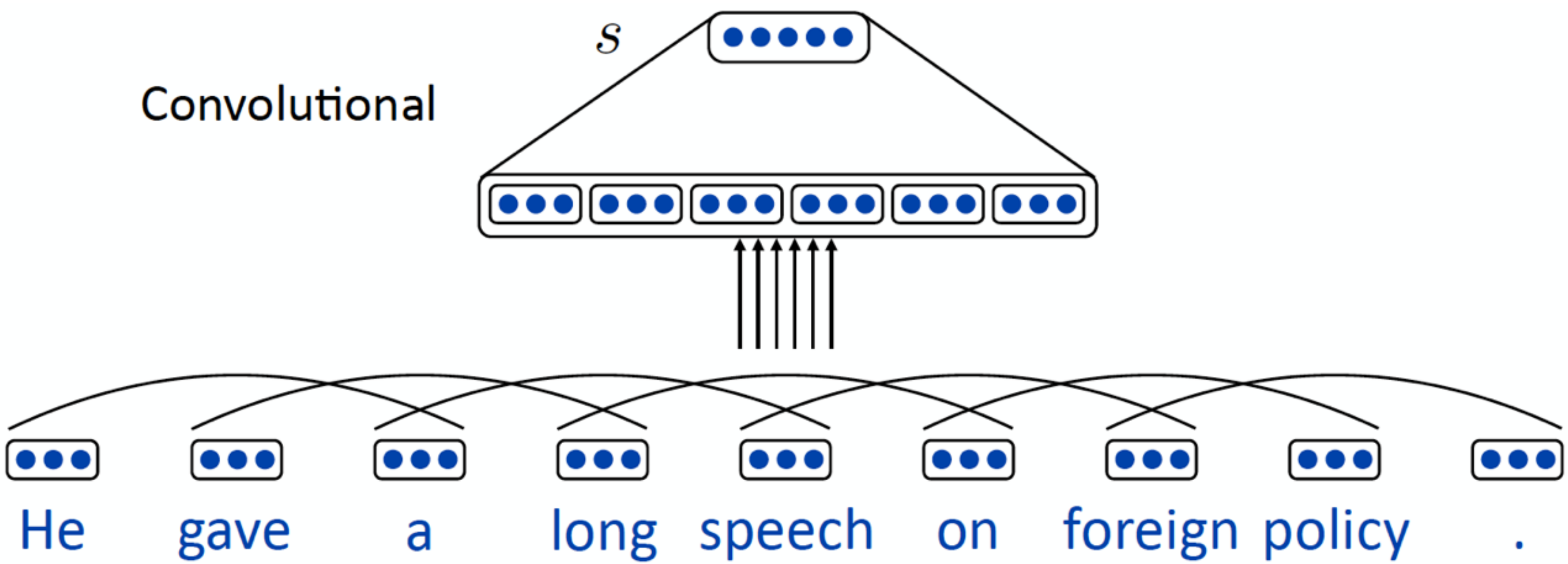
Results



Durrett, G., & Klein, D. (2015). Neural CRF Parsing. ACL.

Neural CRF for Constituency Parsing

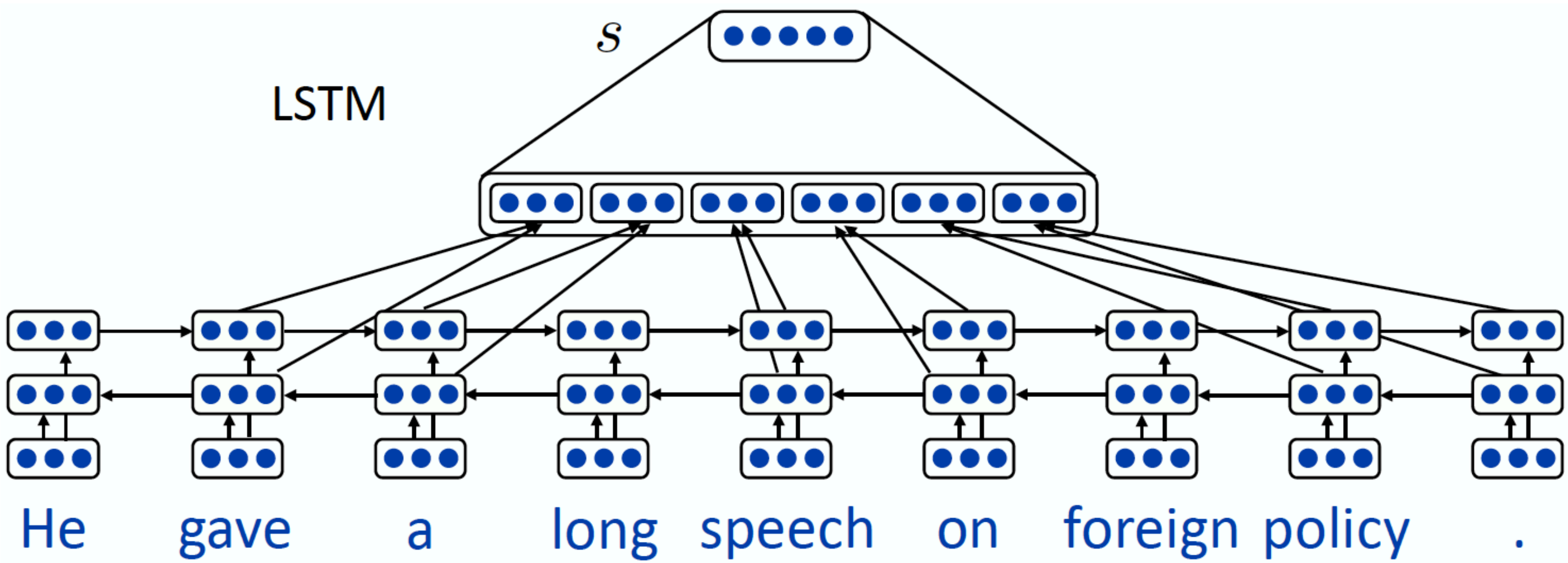
- More neural networks



Durrett, G., & Klein, D. (2015). Neural CRF Parsing. ACL.

Neural CRF for Constituency Parsing

- More neural networks

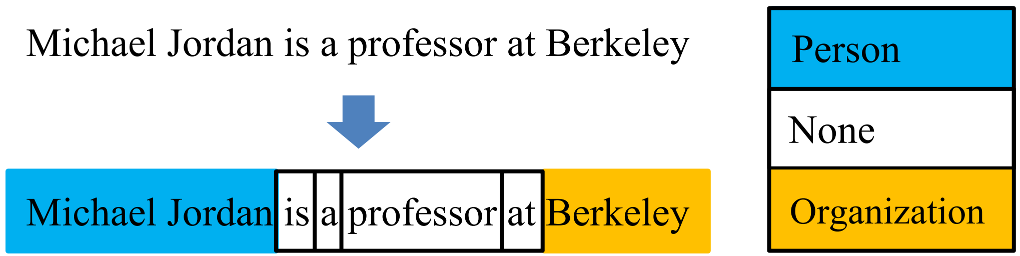


Durrett, G., & Klein, D. (2015). Neural CRF Parsing. ACL.

Part 4.2: Neural Semi-CRF

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

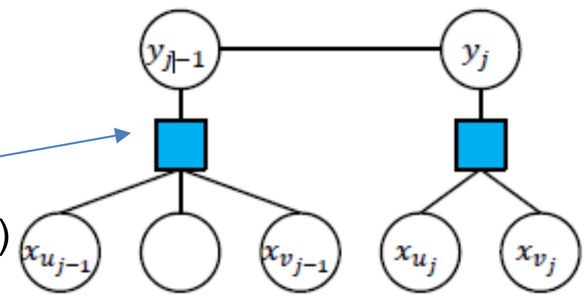
Pudong development and construction

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, Ting Liu. (2016). Exploring Segment Representations for Neural Segmentation Models. IJCAI.

Semi-CRF

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

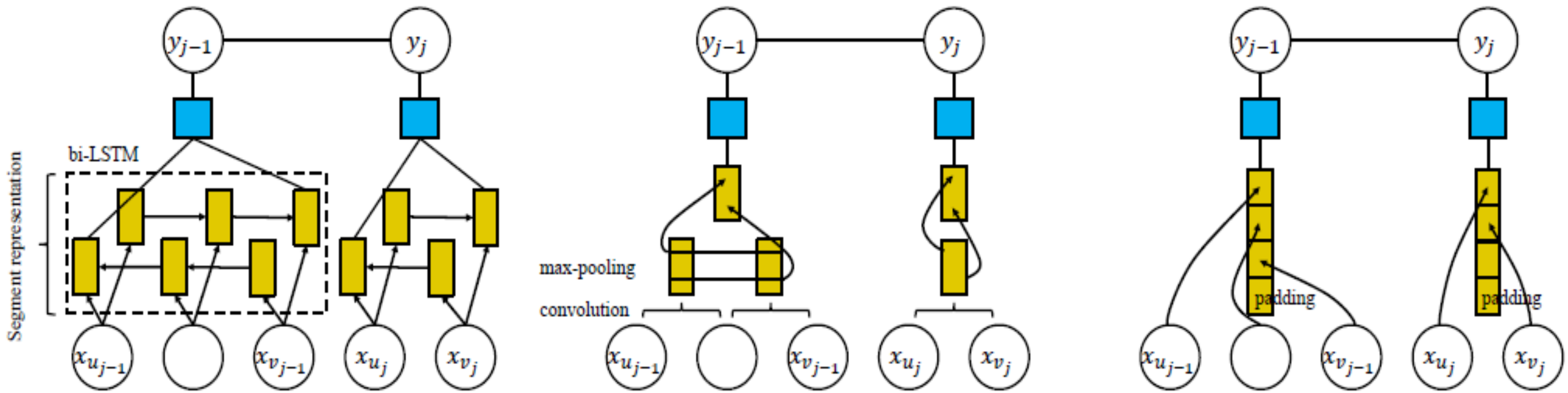
Feature extraction G(x,s)



Can we represent segments with vectors?

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, Ting Liu. (2016). Exploring Segment Representations for Neural Segmentation Models. IJCAI.

Compositional Segment Representation



(b) SRNN (c) SCNN (d) SCONCATE

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, Ting Liu. (2016). Exploring Segment Representations for Neural Segmentation Models. IJCAI.

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:

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

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

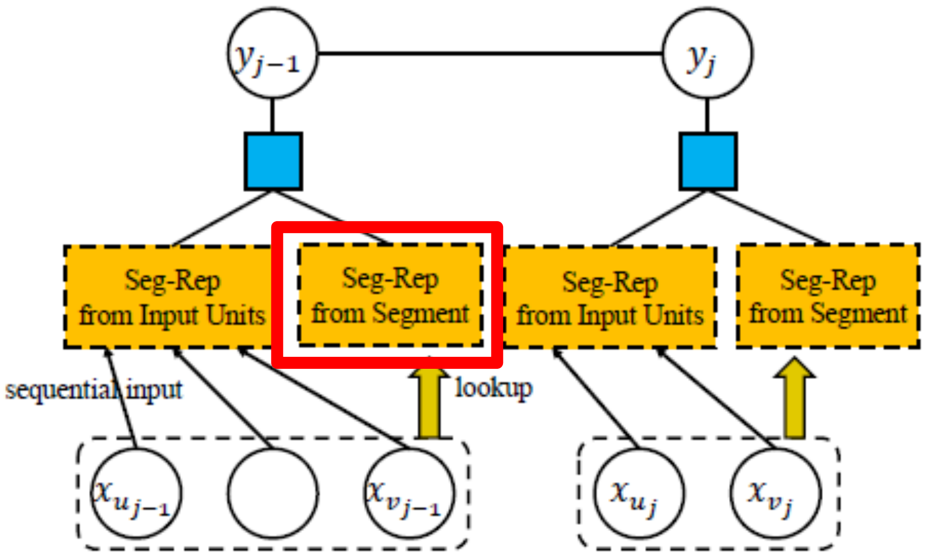
Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, Ting Liu. (2016). Exploring Segment Representations for Neural Segmentation Models. IJCAI.

Results

	model	NER CoNLL03 dev	NER CoNLL03 test	CWS CTB6 dev	CWS CTB6 test	CWS PKU dev	CWS PKU test	CWS MSR dev	CWS MSR test	spd
baseline	NN-LABELER	93.03	88.62	93.70	93.06	93.57	92.99	93.22	93.79	**3.30**
	NN-CRF	**93.06**	**89.08**	94.33	93.65	94.09	93.28	93.81	94.17	2.72
	SPARSE-CRF	88.87	83.43	**95.68**	**95.08**	**95.85**	**95.06**	**96.09**	**96.54**	
neural semi-CRF	SRNN	92.97	88.63	94.56	94.06	94.86	93.91	94.38	95.21	0.62
	SCONCATE	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

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, Ting Liu. (2016). Exploring Segment Representations for Neural Segmentation Models. IJCAI.

Segment-level Representation



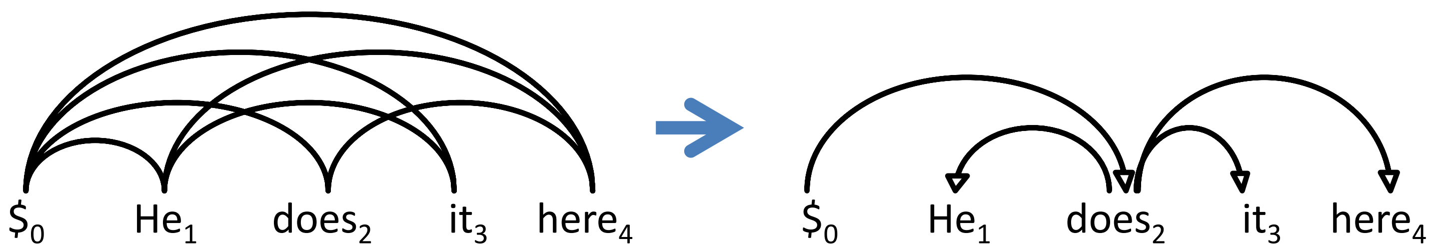
model	CoNLL03	CTB6	PKU	MSR
NN-LABELER	88.62	93.06	92.99	93.79
NN-CRF	89.08	93.65	93.28	94.17
SPARSE-CRF	83.43	95.08	95.06	96.54
SRNN	88.63	94.06	93.91	95.21
+SEMB-HETERO	89.59	**95.48**	95.60	97.39
	+0.96	*+1.42*	*+1.69*	*+2.18*
SCONCATE	89.07	93.96	93.57	94.53
+SEMB-HETERO	**89.77**	95.42	**95.67**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

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, Ting Liu. (2016). Exploring Segment Representations for Neural Segmentation Models. IJCAI.

Part 4.3: Neural Graph-based Parsing

Graph-based Dependency Parsing

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



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

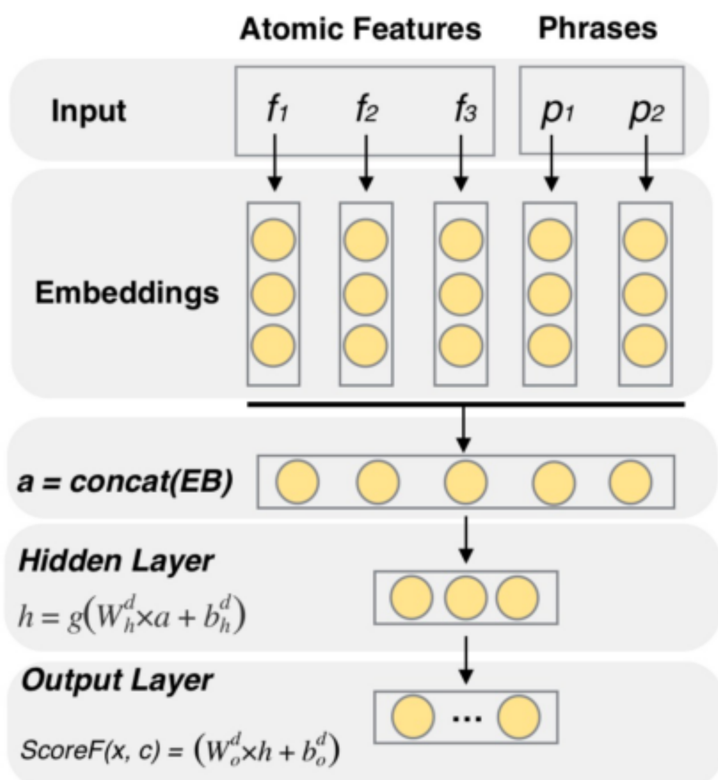
How to Score an Arc?

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

* 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]
[VERB, *, IN, left, 5]	[VERB, JJ, IN, left, 5]	[VERB, JJ, *, left, 5]	[NOUN, *, IN, left, 5]	[NOUN, VERB, IN, left, 5]

NN for Graph-based Parsing



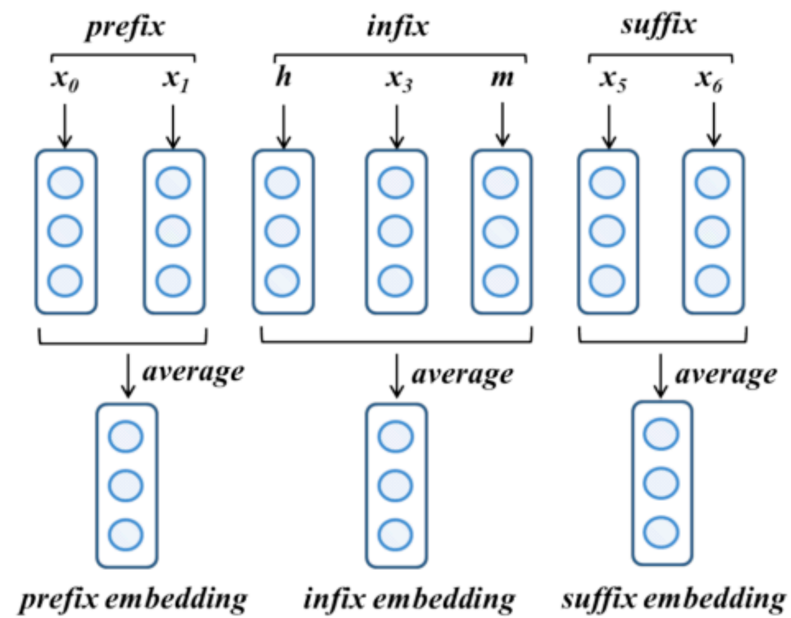


Figure 3: Illustration for phrase embeddings. h, m and x_0 to x_6 are words in the sentence.

Pei, W., Ge, T., & Chang, B. (2015). An Effective Neural Network Model for Graph-based Dependency Parsing. ACL.

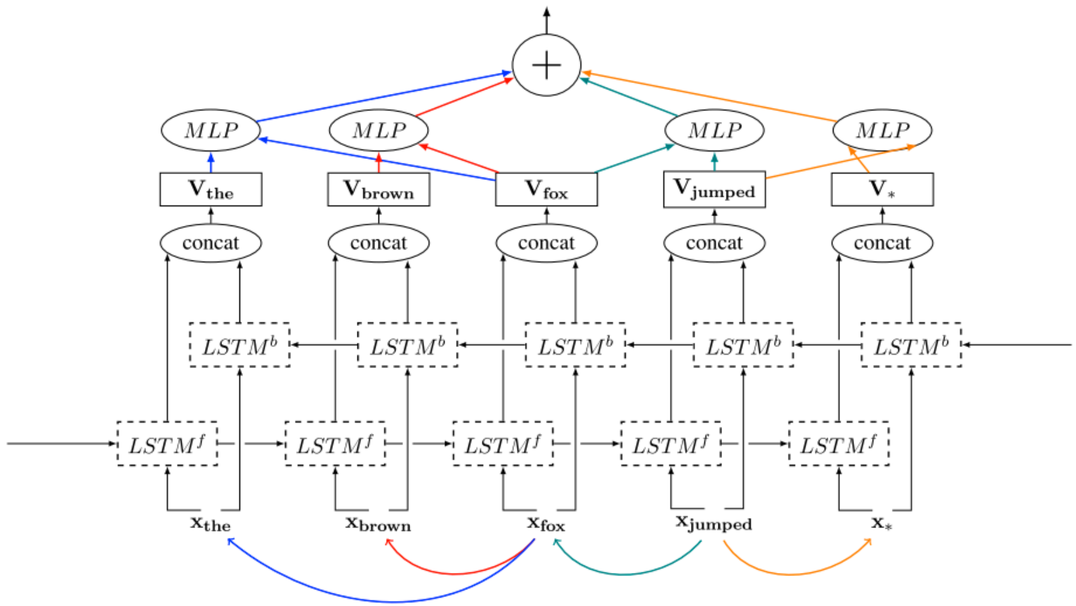
Results

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

Pei, W., Ge, T., & Chang, B. (2015). An Effective Neural Network Model for Graph-based Dependency Parsing. ACL.

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



Kiperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. TACL.

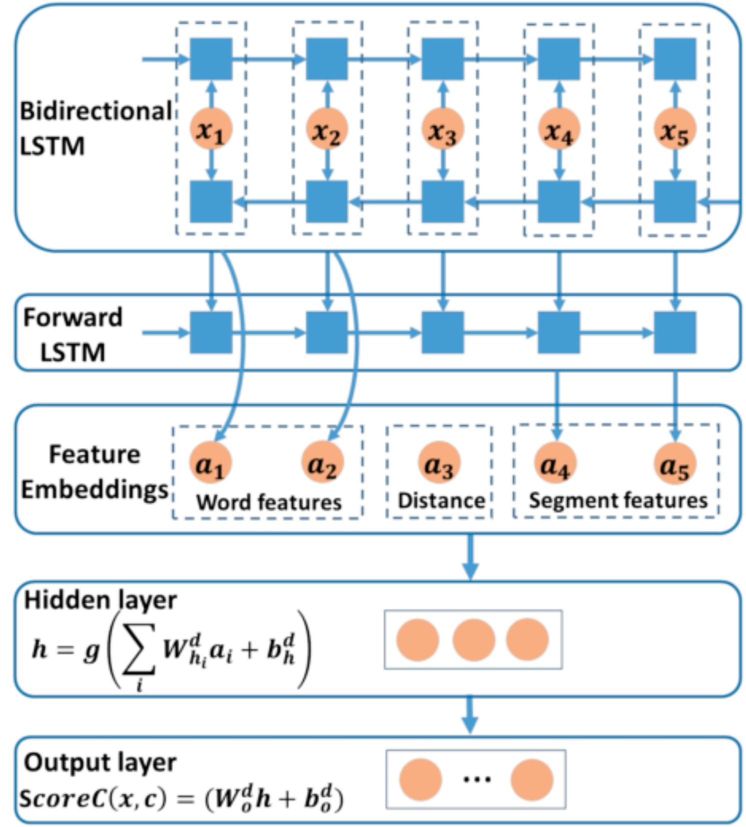
Results

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	**86.6**	**85.1**
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	–	–	**93.2**	**91.2**	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	**87.6**	86.1
Weiss15	transition (greedy)	large feature set (dense)	YES	–	93.2	91.2	–	–
Weiss15	transition (beam)	large feature set (dense)	YES	–	**94.0**	**92.0**	–	–
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	**87.6**	**86.2**
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	–

Kiperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. TACL.

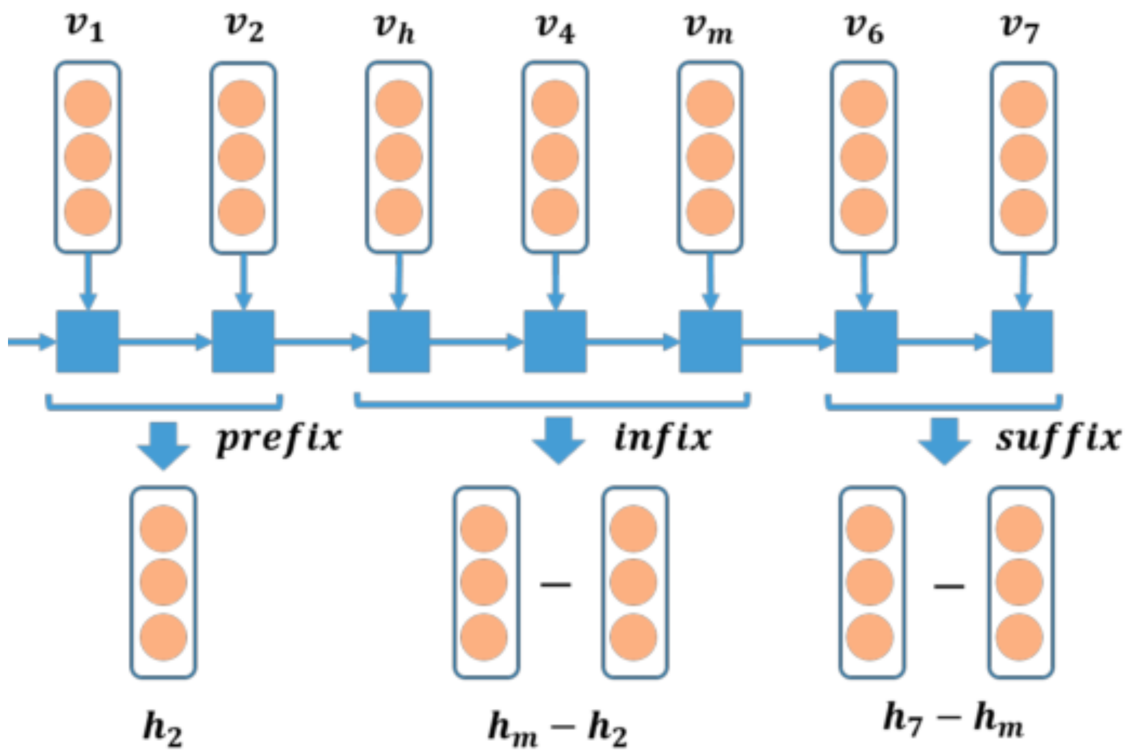
BI-LSTM for Graph-based Parsing-II

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



Wang, W., & Chang, B. (2016). Graph-based Dependency Parsing with Bidirectional LSTM. ACL.

Learning Segment Embeddings



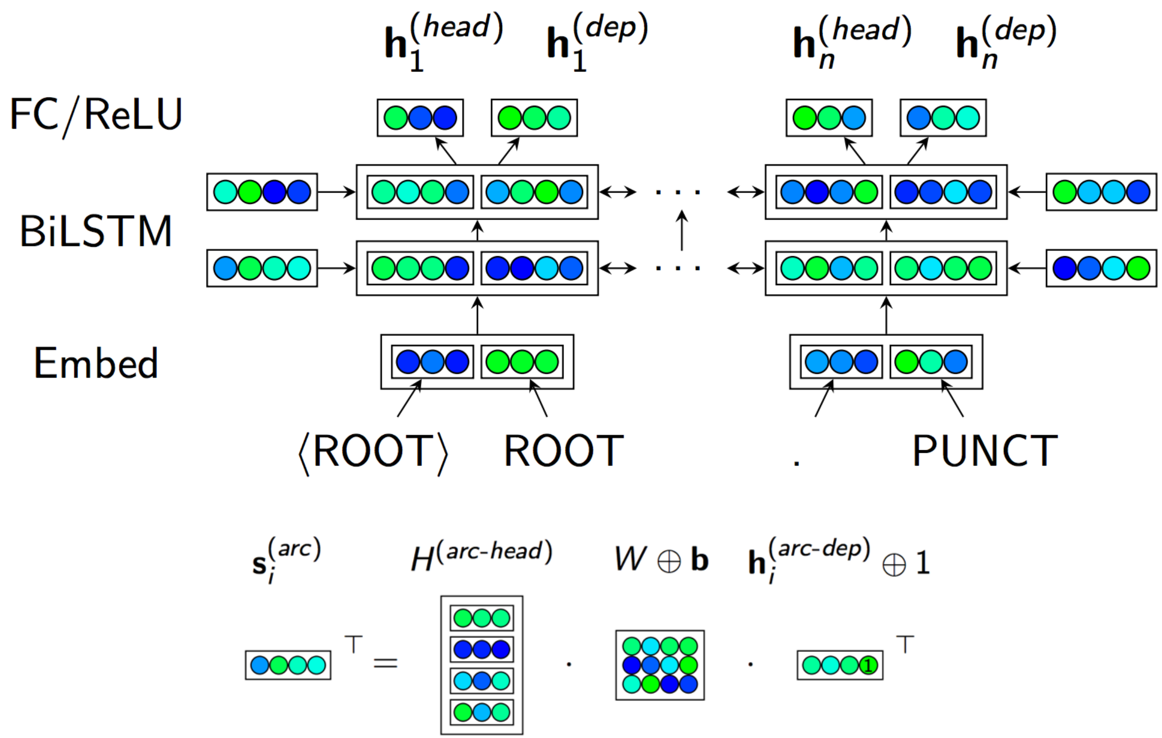
Wang, W., & Chang, B. (2016). Graph-based Dependency Parsing with Bidirectional LSTM. ACL.

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
	Our basic model	93.09	92.03	**61**
	Our basic model + segment	**93.51**	**92.45**	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
	(Zhang and McDonald, 2014)	**93.57**	**92.48**	N/A

Wang, W., & Chang, B. (2016). Graph-based Dependency Parsing with Bidirectional LSTM. ACL.

Deep Biaffine Attention for Dependency Parsing



Timothy Dozat and Christopher D. Manning. Deep Biaffine Attention for Neural Dependency Parsing. ICLR 2017.

Results

Type	Model	English PTB-SD 3.3.0 UAS	English PTB-SD 3.3.0 LAS	Chinese PTB 5.1 UAS	Chinese PTB 5.1 LAS
Transition	Ballesteros et al. (2016)	93.56	91.42	87.65	86.21
	Andor et al. (2016)	94.61	92.79	–	–
	Kuncoro et al. (2016)	**95.8**	**94.6**	–	–
Graph	Kiperwasser & Goldberg (2016)	93.9	91.9	87.6	86.1
	Cheng et al. (2016)	94.10	91.49	88.1	85.7
	Hashimoto et al. (2016)	94.67	92.90	–	–
	Deep Biaffine	95.74	94.08	**89.30**	**88.23**

- Tuning Adam

Model	Adam UAS	LAS
$\beta_2 = .9$	**95.75**	**94.22**
$\beta_2 = .999$	95.53*	93.91*

Conclusion

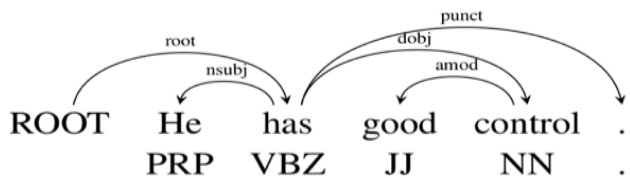
- Neural nets can provide continuous features in discrete structured models
- Inference and learning are almost unchanged from the purely discrete model

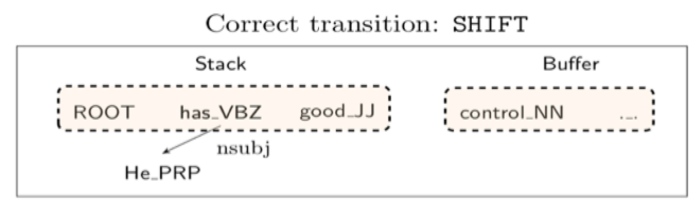
Part 5: Neural Transition-based Methods

Part 5.1: Neural Transition-based Dependency Parsing with Greedy Search

Dependency Parsing

- Neural MaltParser





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

Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Network. ACL.

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

Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Network. ACL.

Dependency Parsing

Softmax layer:

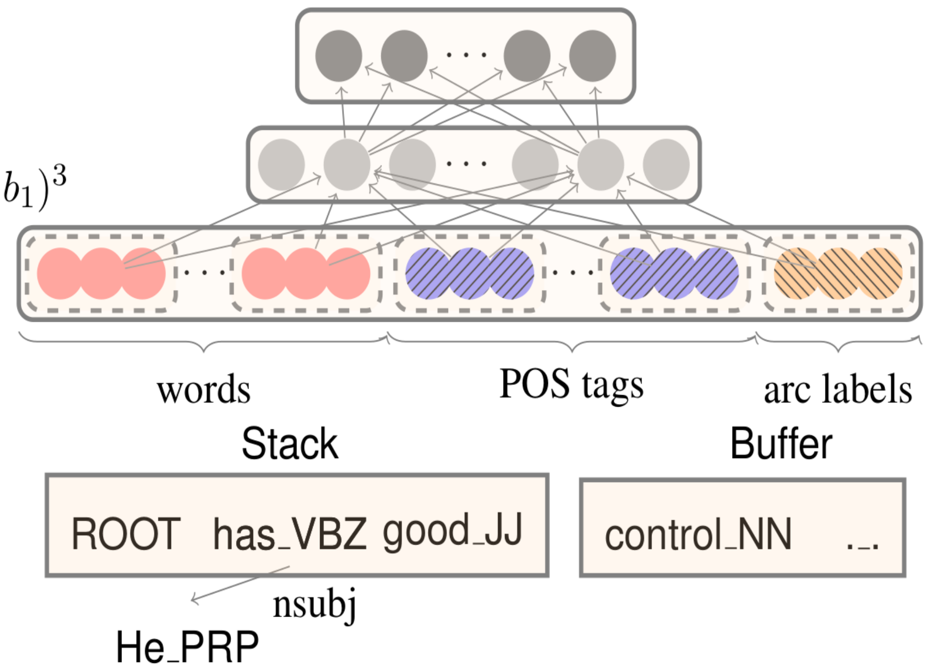
$$p = \mathtt{softmax}(W_2 h)$$

Hidden layer:

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

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

Configuration



Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Network. ACL.

Dependency Parsing

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
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	**92.0**	**89.7**	**91.8**	**89.6**	**654**

PTB (SD)

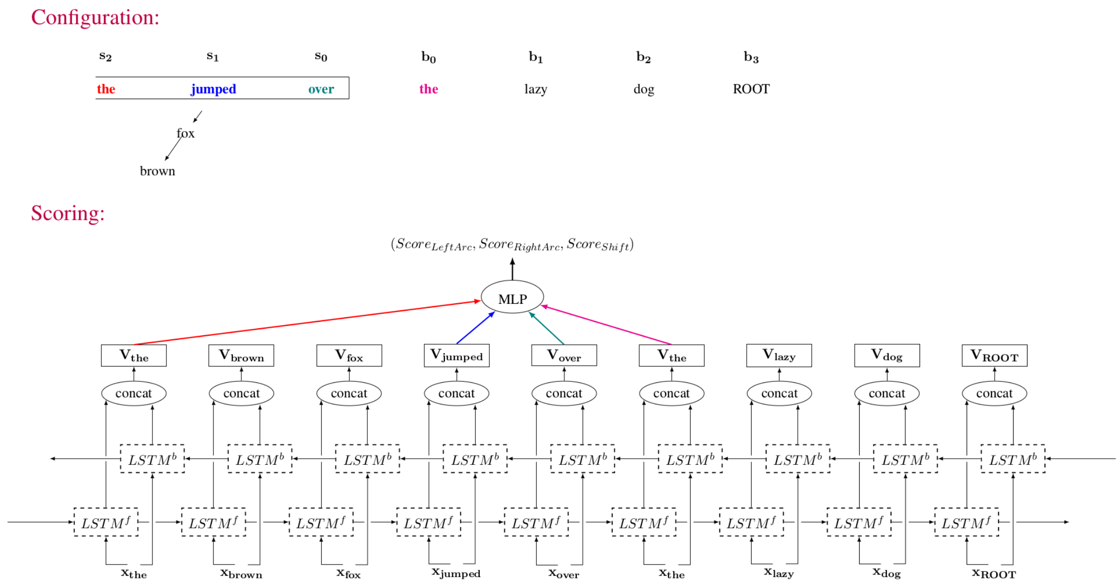
Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	**84.0**	82.1	83.0	81.2	6
Our parser	**84.0**	**82.4**	**83.9**	**82.4**	**936**

CTB (SD)

Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Network. ACL.

Dependency Parsing

- Chen and Manning with richer (LSTM) features



Keperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. TACL.

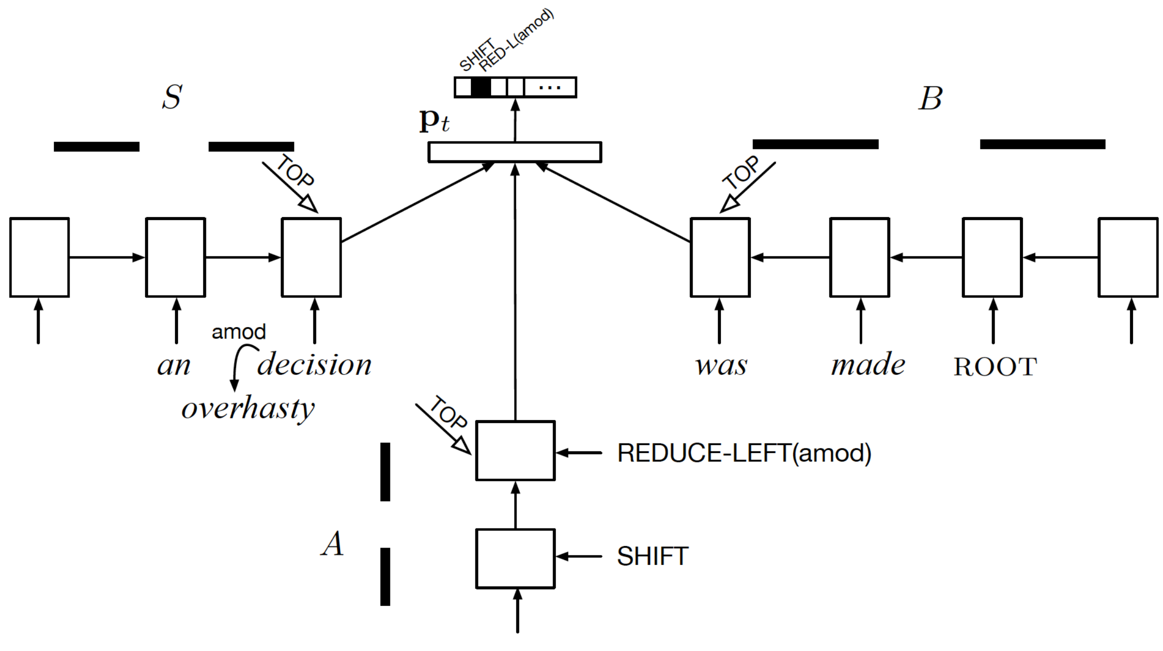
Dependency Parsing

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	**86.6**	**85.1**
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	–	–	**93.2**	**91.2**	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	**87.6**	86.1
Weiss15	transition (greedy)	large feature set (dense)	YES	–	93.2	91.2	–	–
Weiss15	transition (beam)	large feature set (dense)	YES	–	**94.0**	**92.0**	–	–
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	**87.6**	**86.2**
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	–

Keperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. TACL.

Dependency Parsing

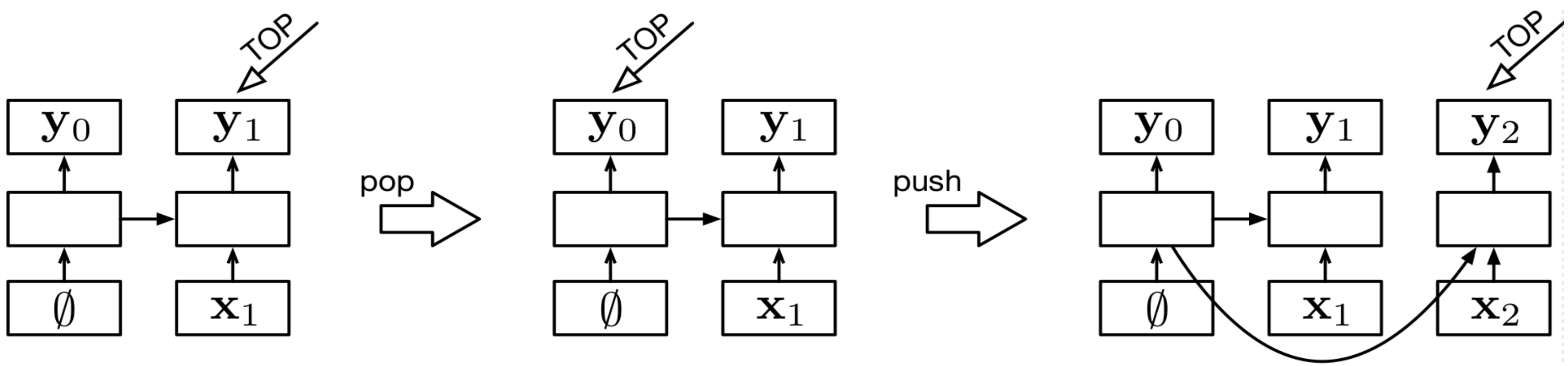
- Dyer Parser (Chen and Manning with less features)



Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-Based Dependency Parsing with Stack Long Short-Term Memory. ACL.

Dependency Parsing

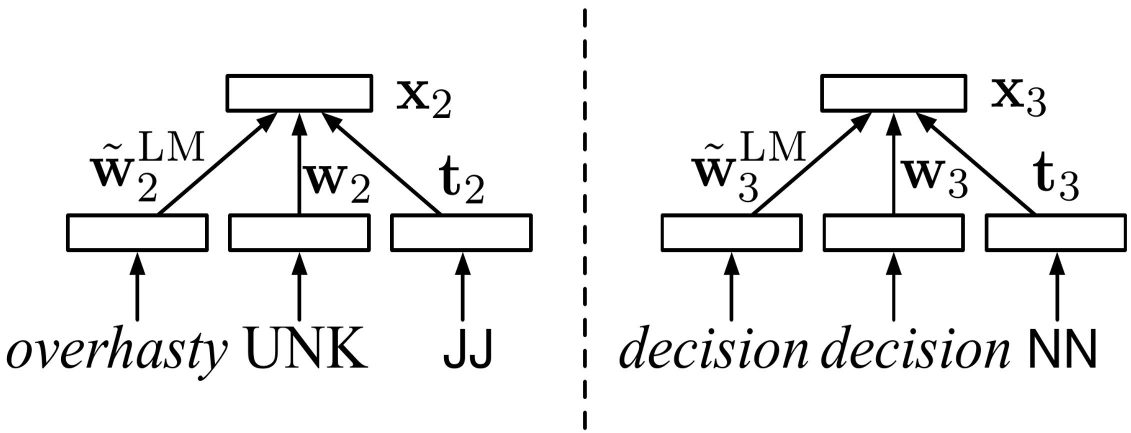
- Stack LSTM



Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-Based Dependency Parsing with Stack Long Short-Term Memory. ACL.

Dependency Parsing

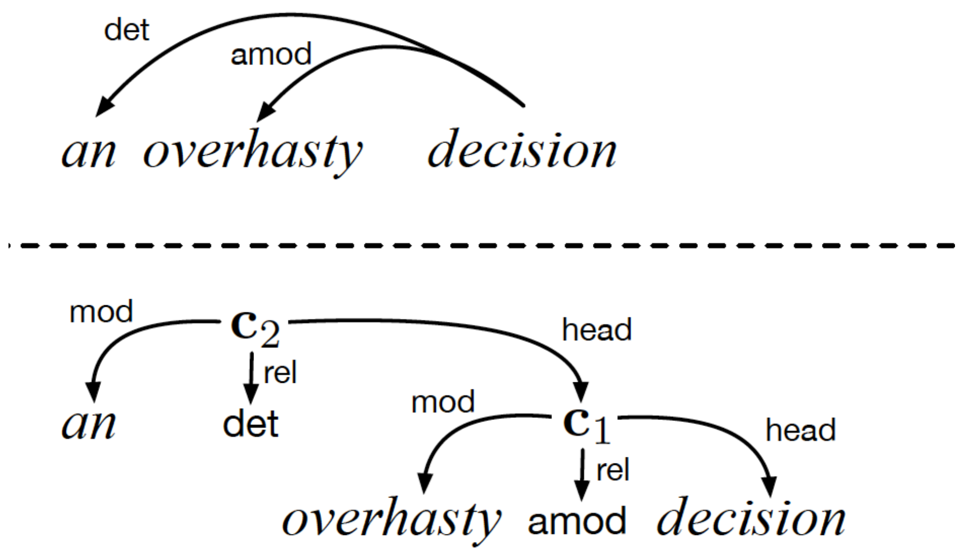
- Two types of word embeddings



Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-Based Dependency Parsing with Stack Long Short-Term Memory. ACL.

Dependency Parsing

- Subtree Representation (Recursive NN)



Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-Based Dependency Parsing with Stack Long Short-Term Memory. ACL.

Dependency Parsing

	Development		Test	
	UAS	LAS	UAS	LAS
S-LSTM	**93.2**	**90.9**	**93.1**	**90.9**
−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

PTB (SD)

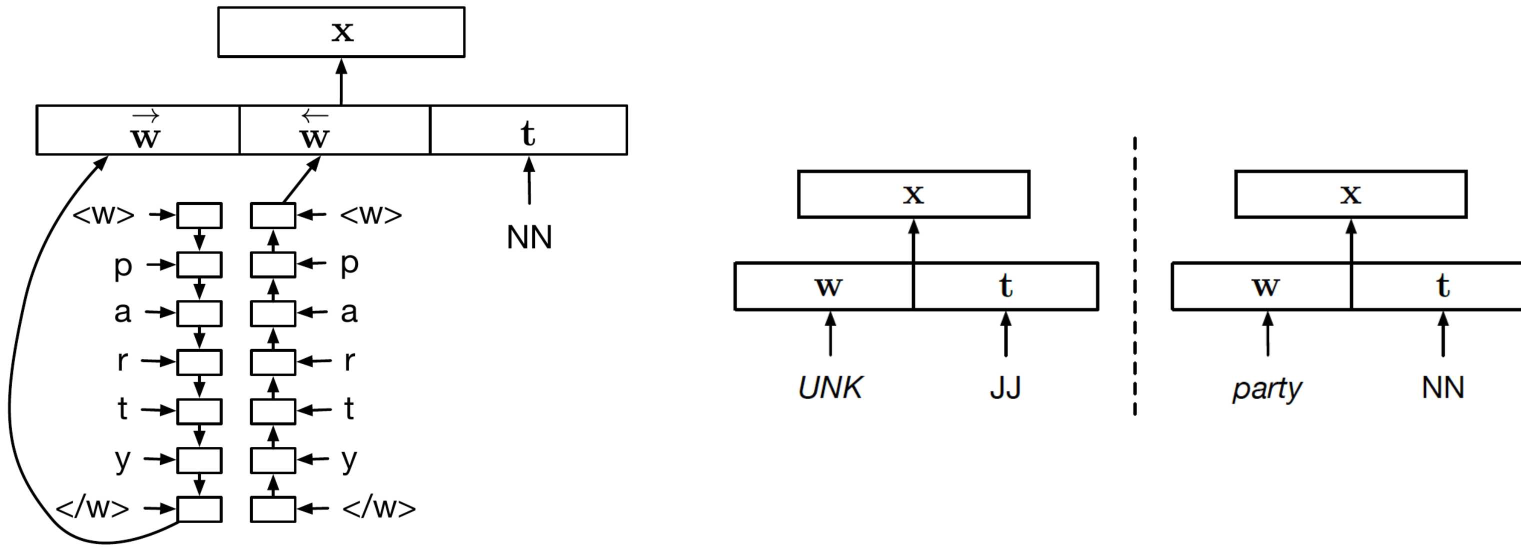
	Dev. set		Test set	
	UAS	LAS	UAS	LAS
S-LSTM	**87.2**	**85.9**	**87.2**	**85.7**
−composition	85.8	84.0	85.3	83.6
−pretraining	86.3	84.7	85.7	84.1
−POS	82.8	79.8	82.2	79.1
S-RNN	86.3	84.7	86.1	84.6
C&M (2014)	84.0	82.4	83.9	82.4

CTB (CTB5)

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-Based Dependency Parsing with Stack Long Short-Term Memory. ACL.

Dependency Parsing

- Dyer et al. with character based word vector



Ballesteros, M., Dyer, C., & Smith, N. A. (2015). Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. EMNLP.

Dependency Parsing

UAS

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

LAS

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

Ballesteros, M., Dyer, C., & Smith, N. A. (2015). Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. EMNLP.

Part 5.2: Neural Transition-based Dependency Parsing with Beam Search

Dependency Parsing

- Sentence-level log likelihood

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

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

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

$$L(\theta) = - \sum_{(x_i, y_i) \in (X,Y)} \log p(y_i \mid x_i, \theta)$$

$$= - \sum_{(x_i, y_i) \in (X,Y)} \log \frac{e^{f(x_i,\theta)_i}}{Z(x_i, \theta)}$$

$$= \sum_{(x_i, y_i) \in (X,Y)} \log Z(x_i, \theta) - f(x_i, \theta)_i$$

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

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

$$L'(\theta) = -\sum_{(x_i, y_i) \in (X,Y)} \log p'(y_i \mid x_i, \theta)$$

$$= -\sum_{(x_i, y_i) \in (X,Y)} \log \frac{e^{f(x_i,\theta)_i}}{Z'(x_i, \theta)}$$

$$= \sum_{(x_i, y_i) \in (X,Y)} \log Z'(x_i, \theta) - f(x_i, \theta)_i$$

$$Z'(x, \theta) = \sum_{\boxed{y_j \in \mathrm{BEAM}(x)}} e^{f(x, \theta)_j}$$

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

Dependency Parsing

- 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

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

Dependency Parsing

- Results

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

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

Dependency Parsing

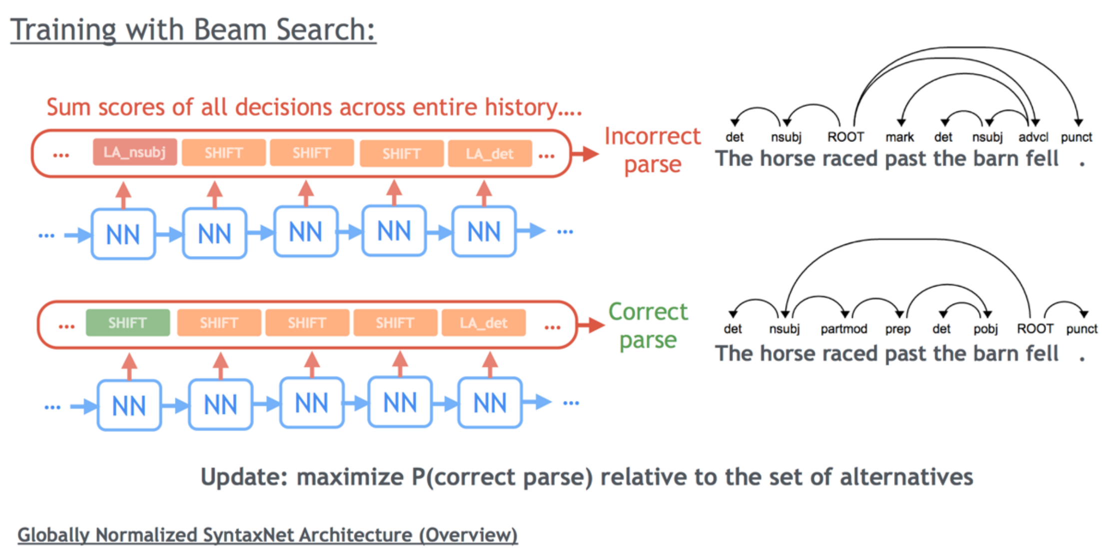
- Results

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	**93.28**	**92.35**	0.07
100	64	93.20	92.27	0.04
100	16	92.40	91.95	0.01

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

Google's SyntaxNet

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



Andor, D., Alberti, Chris., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., & Collins, M. (2016). Globally Normalized Transition-Based Neural Networks. ACL.

Google's SyntaxNet

Method	WSJ UAS	WSJ LAS	Union-News UAS	Union-News LAS	Union-Web UAS	Union-Web LAS	Union-QTB UAS	Union-QTB 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)	**94.61**	**92.79**	**94.44**	**92.93**	**90.17**	**87.54**	**95.40**	**93.64**
Parsey McParseface (B=8)	-	-	94.15	92.51	89.08	86.29	94.77	93.17

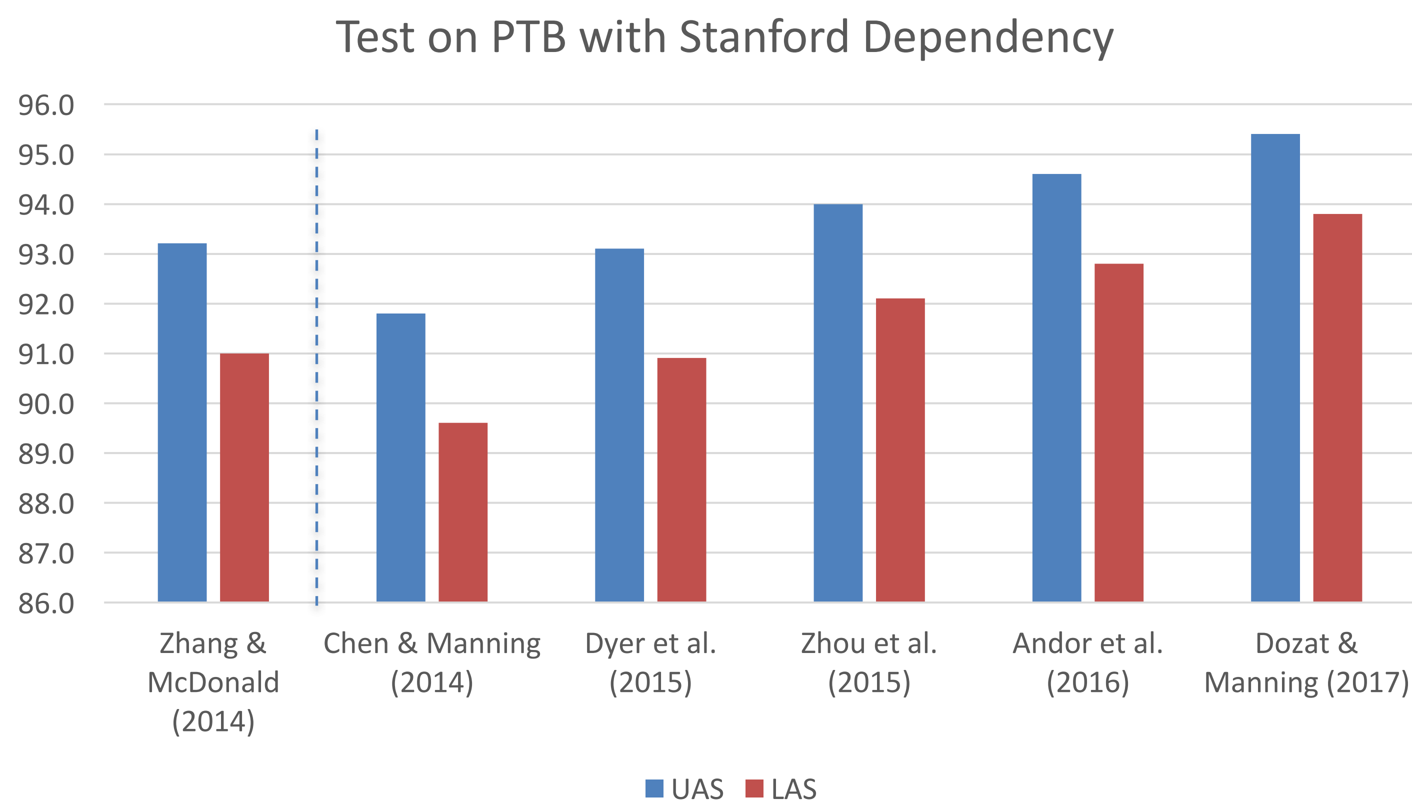
Andor, D., Alberti, Chris., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., & Collins, M. (2016). Globally Normalized Transition-Based Neural Networks. ACL.

Google's SyntaxNet

Method	Catalan		Chinese		Czech		English		German		Japanese		Spanish	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	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	**94.04**	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	**91.37**	**89.38**	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	**93.10**	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)	**92.67**	**89.83**	**84.72**	**80.85**	**88.94**	**84.56**	**93.22**	**91.23**	90.91	89.15	93.65	92.84	**92.62**	**89.95**

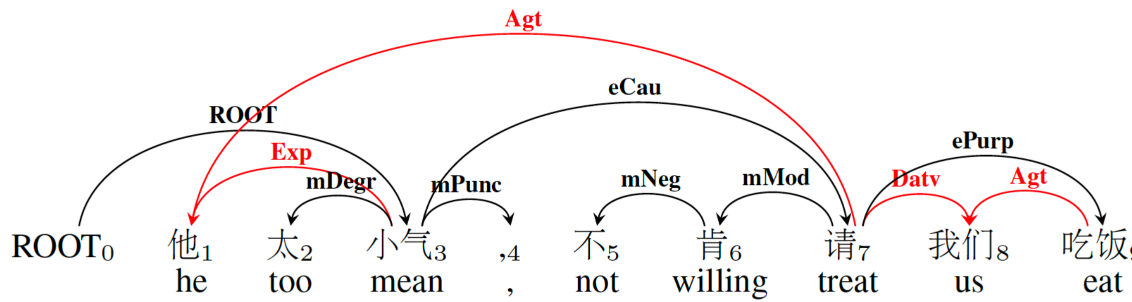
Andor, D., Alberti, Chris., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., & Collins, M. (2016). Globally Normalized Transition-Based Neural Networks. ACL.

Changes of Performance



Part 5.3: Advanced Topics

Semantic Dependency Graph



		Train	Dev	Test
NEWS	#sent	8,301	534	1,233
	#word	250,249	15,325	34,305
TEXT	#sent	10,817	1,546	3,096
	#word	128,095	18,257	36,097

Wanxiang Che, Yu Ding, Yanqiu Shao, Ting Liu. **SemEval-2016 Task 9**: Chinese Semantic Dependency Parsing.

Semantic Dependency Graph

- List-based transition system

Stack s
Processed words

Stack p
Skipped words

Buffer b
Unprocessed words

- New transition actions
  - Left-Reduce, Right-Shift, No-Shift, No-Reduce, Left-Pass, Right-Pass, No-Pass

Yuxuan Wang, Jiang Guo, Wanxiang Che and Ting Liu. Transition-Based Chinese Semantic Dependency Graph Parsing. CCL 2016. **Best Paper Award.**

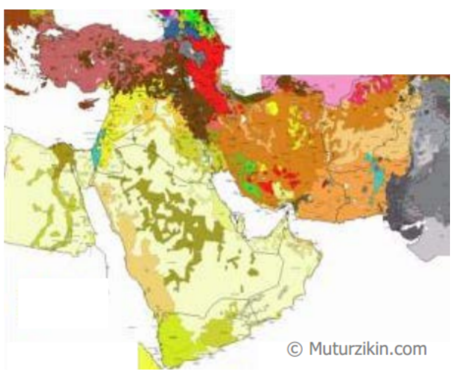
Semantic Dependency Graph

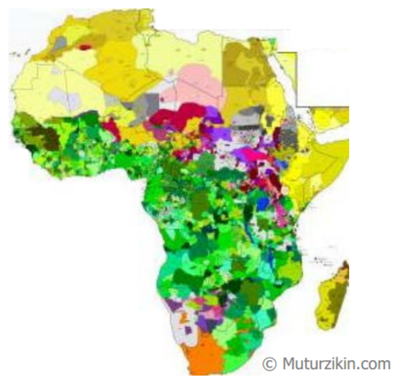
- Results

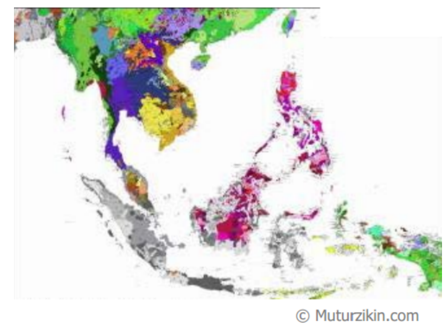
System	NEWS				TEXTBOOKS			
	LF	UF	NLF	NUF	LF	UF	NLF	NUF
IHS-RD-Belarus	59.06	77.64	40.84	60.20	68.59	82.41	50.57	64.58
OCLSP (lbpg)	57.22	74.93	45.57	58.03	65.54	79.39	51.75	63.21
OCLSP (lbpgs)	57.81	75.54	41.56	54.34	66.21	79.85	47.79	55.51
OCLSP (lbpg75)	57.78	75.40	48.89	58.28	66.38	79.91	57.51	63.87
OSU_CHGCG	55.69	73.72	49.23	60.71	65.17	78.83	54.70	65.71
1-LSTM-Basic	62.23	80.42	49.18	63.90	71.51	84.95	59.70	71.63
1-LSTM-Bi-Tree	**63.08**	**80.90**	**52.81**	**67.08**	**72.54**	**85.47**	**60.83**	**72.47**

Multilingual Dependency Parsing

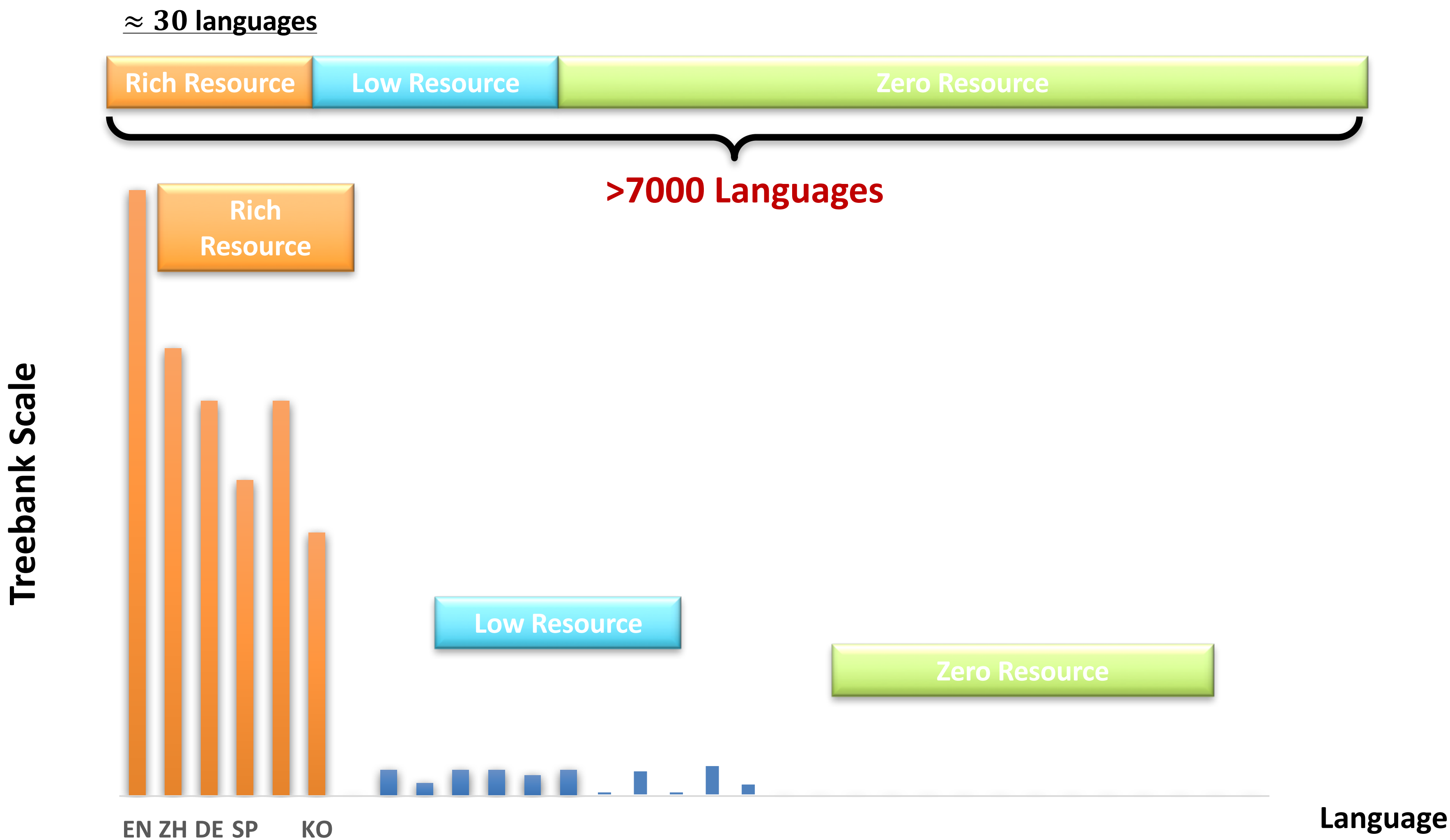
- Over 7,000 languages all around the world
  - Most of the languages are ***low-resource*** for dependency parsing

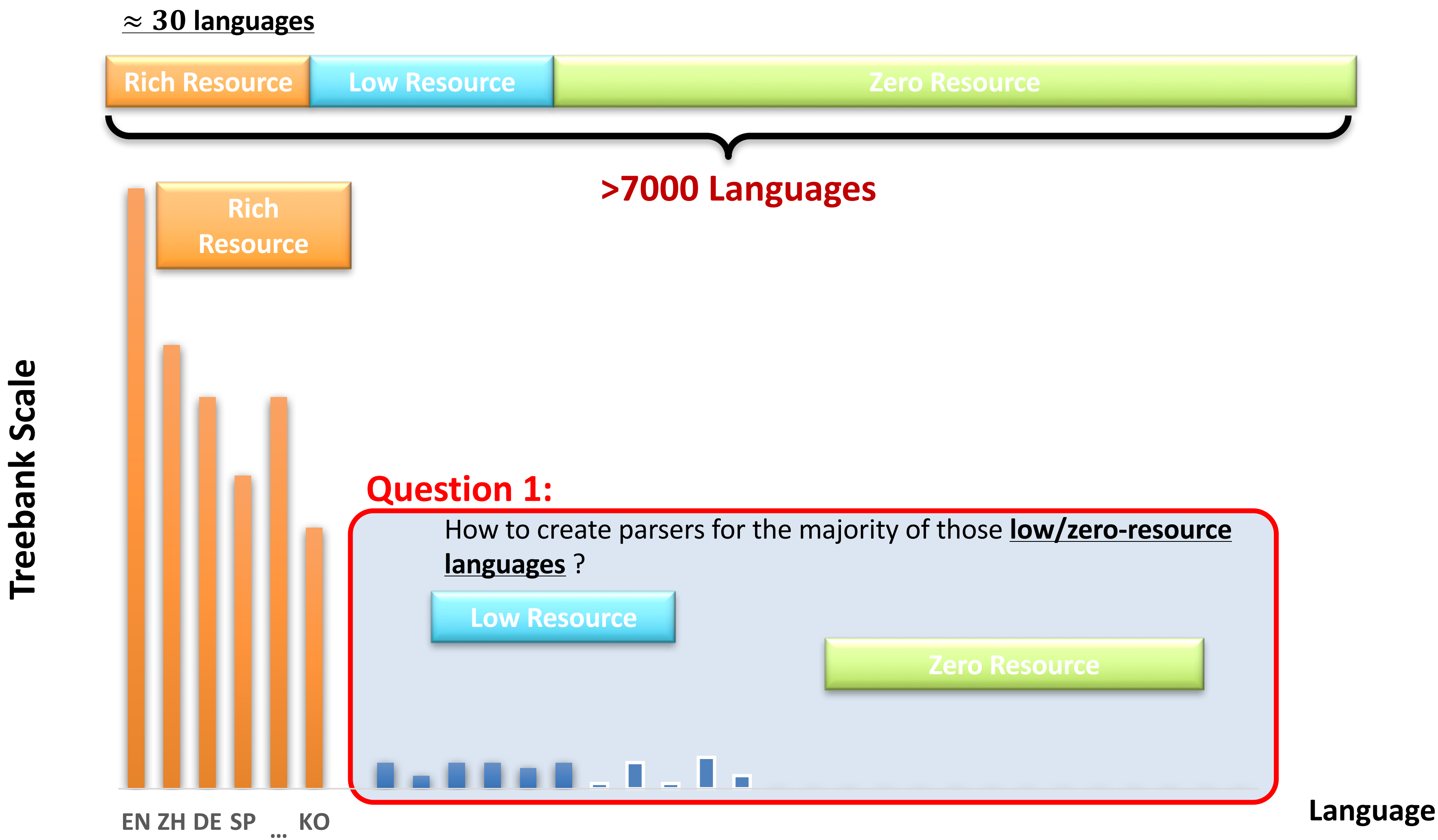


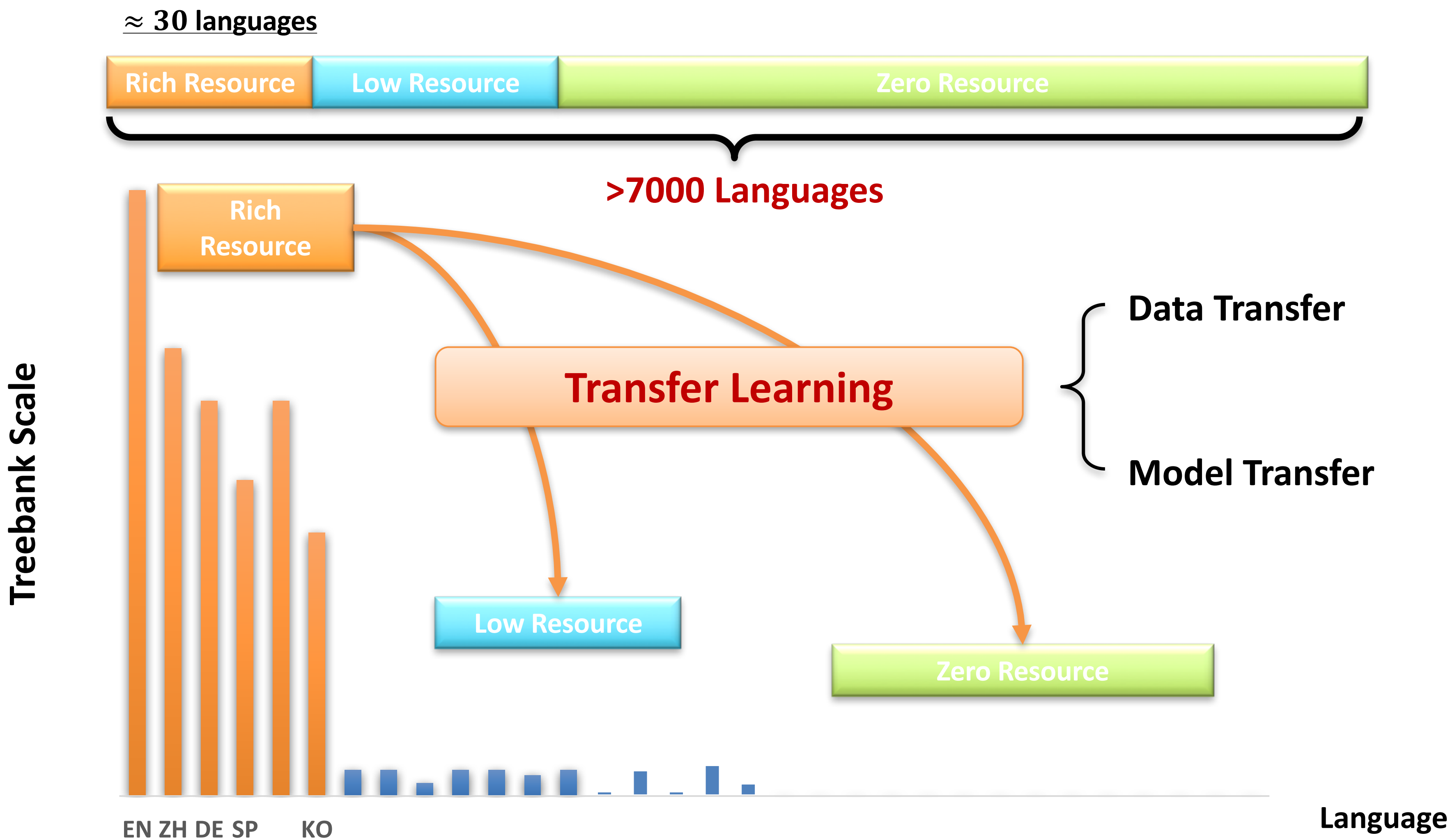




(Colors indicate language families)


≈ 30 languages
Rich Resource
Low Resource
Zero Resource
>7000 Languages
Rich Resource
Low Resource
Zero Resource
Treebank Scale
EN ZH DE SP … KO
Language

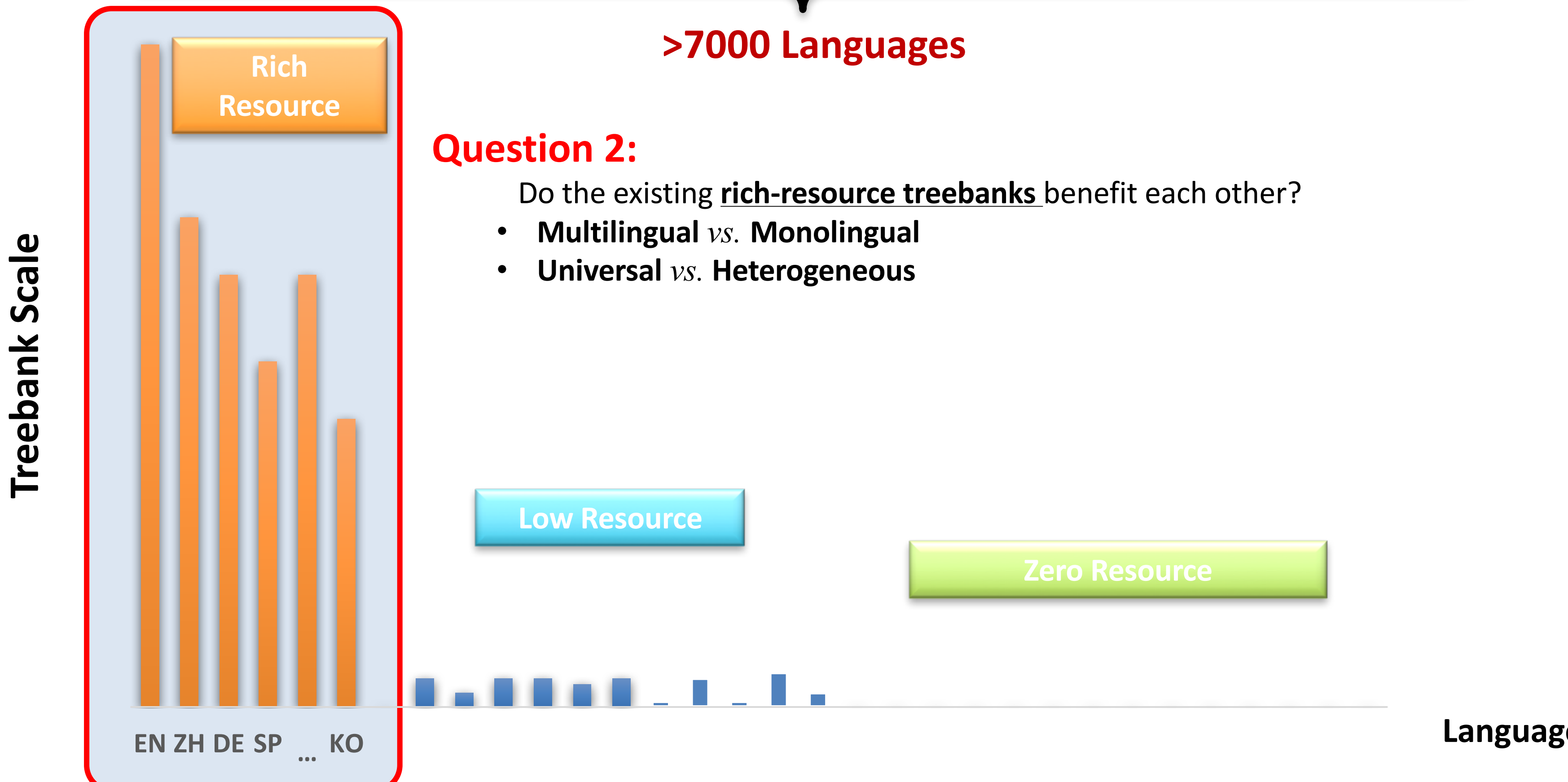

≈ 30 languages
Rich Resource
Low Resource
Zero Resource
>7000 Languages
Rich Resource
Treebank Scale
Question 1:
How to create parsers for the majority of those **low/zero-resource languages** ?
Low Resource
Zero Resource
EN ZH DE SP … KO
Language


≈ 30 languages
Rich Resource
Low Resource
Zero Resource
>7000 Languages
Rich Resource
Transfer Learning
Data Transfer
Model Transfer
Low Resource
Zero Resource
Treebank Scale
Language
EN ZH DE SP … KO

Question 2:

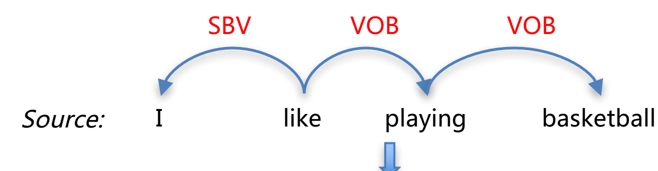
Do the existing **rich-resource treebanks** benefit each other?

- **Multilingual** *vs.* **Monolingual**
- **Universal** *vs.* **Heterogeneous**



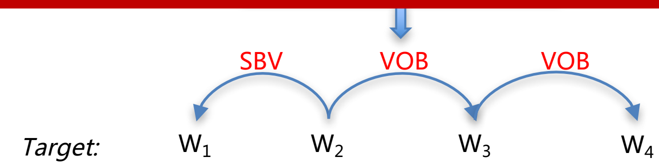
Cross-lingual Dependency Parsing

- Use the model trained on source language to parse the target language



Rich resource language

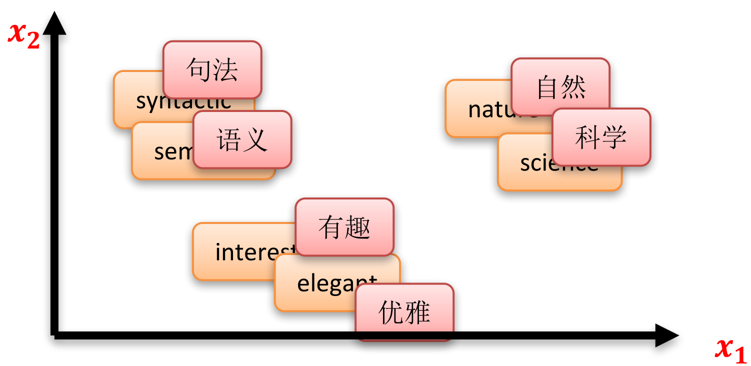
How to overcome the lexical inconsistency problem?

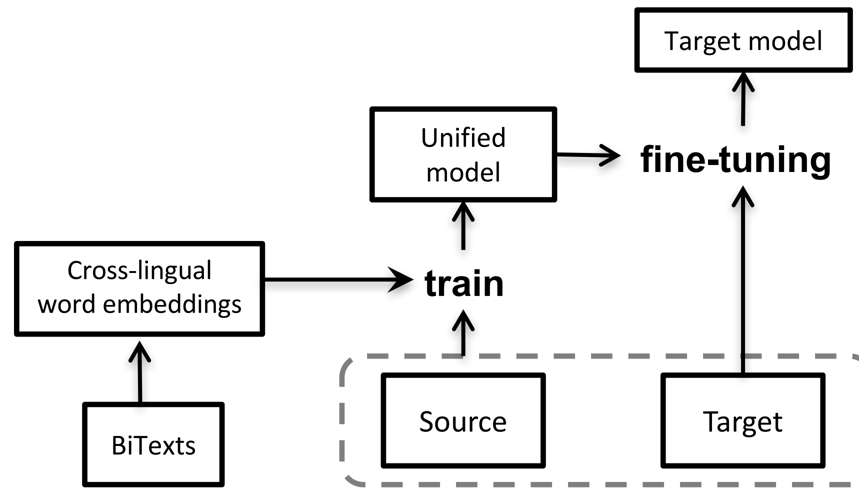


Low resource lanaguage

Cross-lingual Dependency Parsing

- Use bi-lingual word embeddings to overcome the lexical inconsistency problem



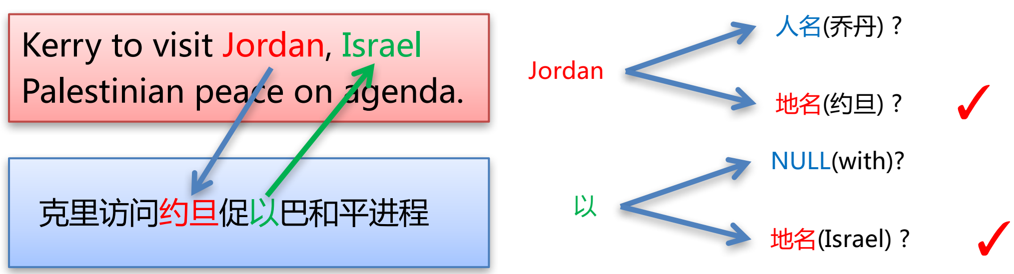


- The performance of target language can be improved more than 4%

Our paper: ACL 2015， AAAI 2016，JAIR 2016，CoNLL 2017

Bi-lingual based Named Entity Recognition

- The parallel corpus have inter-translated named entities

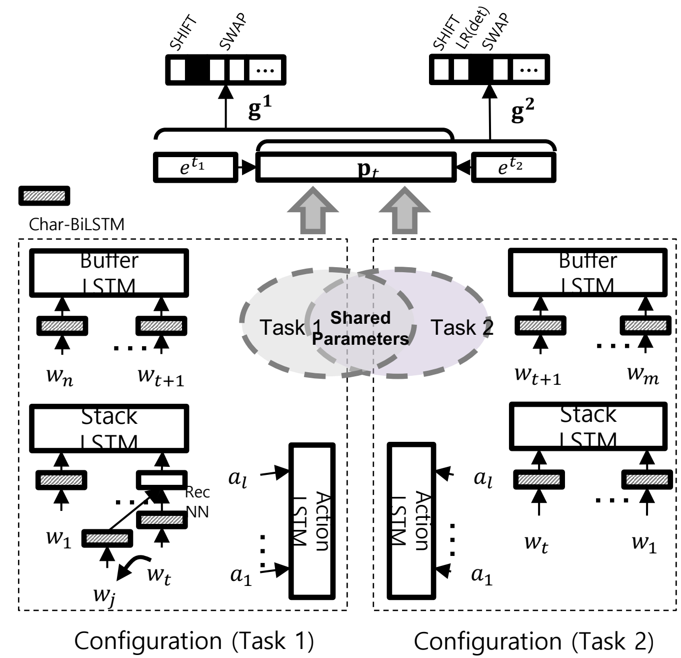


- Bi-lingual constraint based methods

Our paper: NAACL 2013、ACL 2013、AAAI 2013 (Outstanding mention award)

Deep Multi-task Learning Framework

- Each corpus can be looked as a task
  - Multi-lingual treebanks
  - Mono-lingual heterogeneous treebanks
  - Multiple NLP tasks
- **Shared parameters**
  - LSTM(B), LSTM(S)
  - LSTM(A)
  - BiLSTM(chars)
  - RecNN
  - W_A, W_B, W_S
  - $E_{pos}, E_{char}, E_{rel}, E_{act}$



Jiang Guo, Wanxiang Che, Haifeng Wang and Ting Liu. A Universal Framework for Transfer Parsing across Multi-typed Treebanks. Coling 2016.

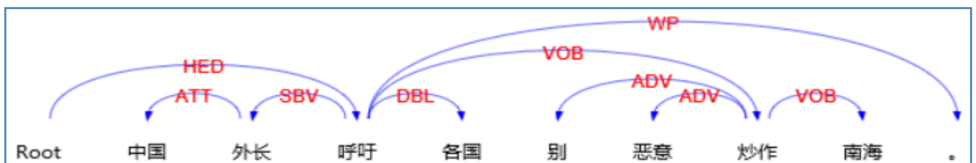
Conclusion

- Neural Transition-based Dependency Parsing
  - Greedy decoding
  - Beam-search decoding
- Advanced Topics
  - Semantic dependency graph parsing
  - Multilingual dependency parsing

Part 6: Applications

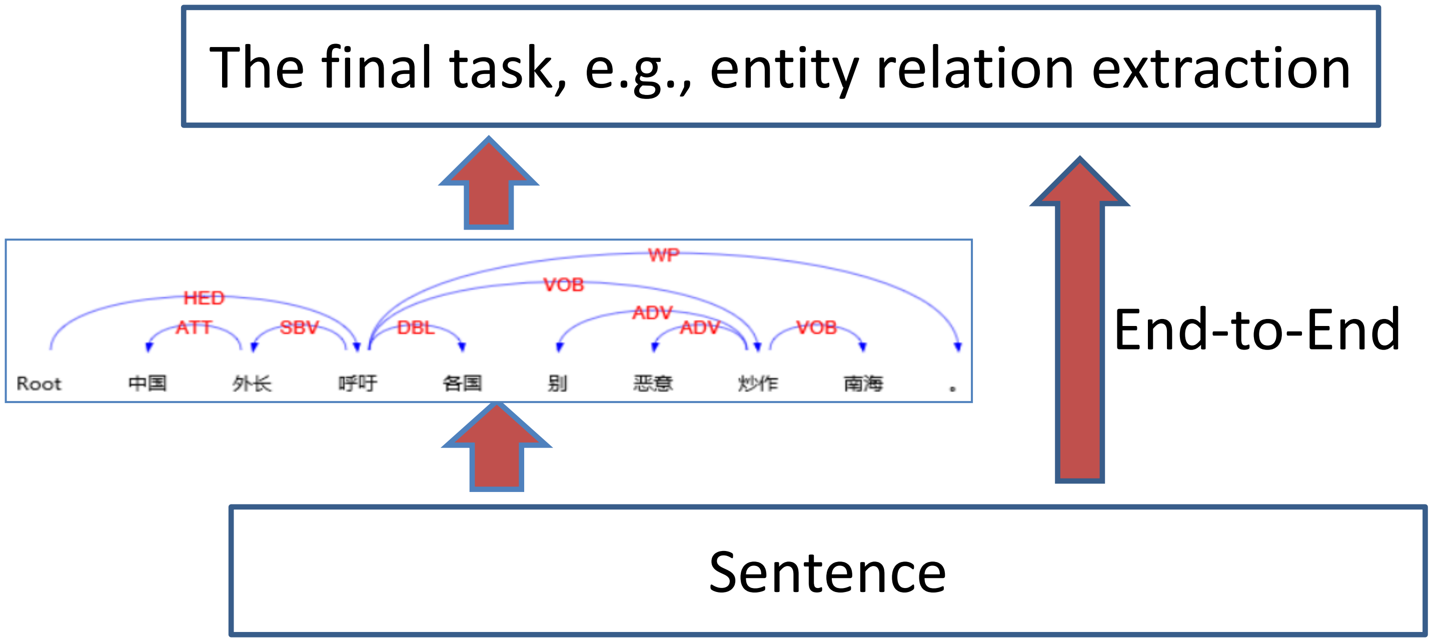
Shallow Learning

The final task, e.g., entity relation extraction



Sentence

Deep Learning

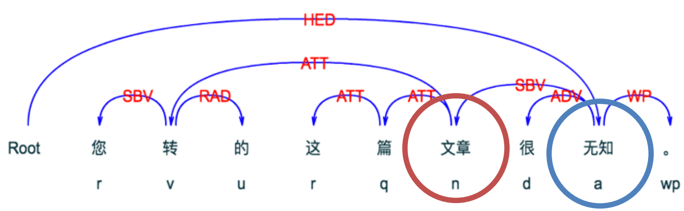


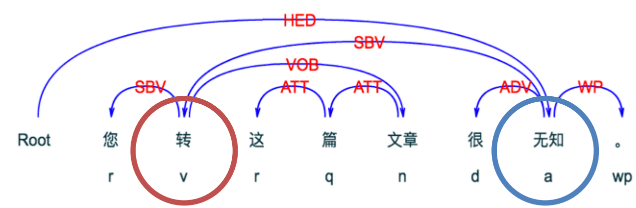
How to Use Tree or Graph Structures?

- As Information Extraction Rules
- As Input Features
- As Input Structures
- As Structured Prediction

As Information Extraction Rules

- For example
  - Polarity-target pair extraction

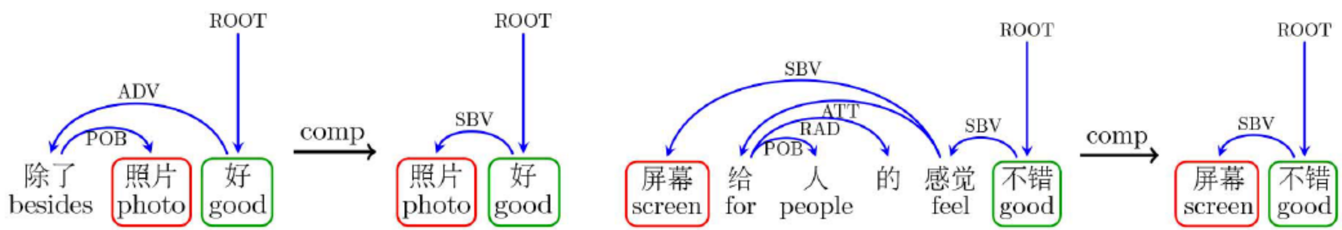




- Problem
  - The extraction rules are very complex
  - The parsing results are inexact

As Information Extraction Rules

- Sentence compression based PT pair extraction
  - Simplify the extraction rules
  - Improve the parsing accuracy

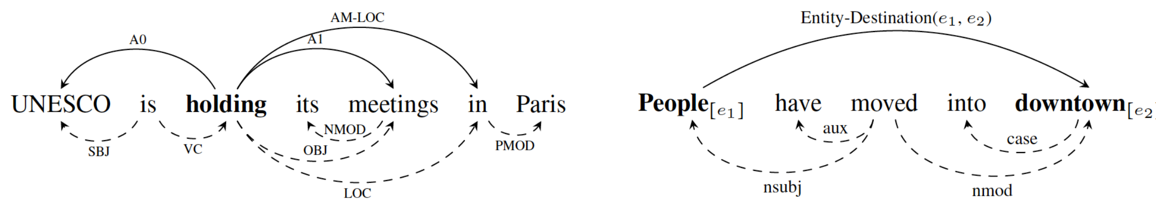


- Use a sequence labeling model to compress sentences
- The PT pair extraction performance improves 3%

Wanxiang Che, Yanyan Zhao, Honglei Guo, Zhong Su, Ting Liu. Sentence Compression for Aspect-Based Sentiment Analysis. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2015, 23(12)

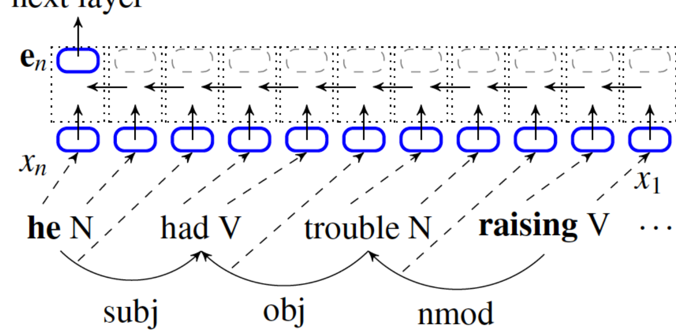
As Input Features

- For Semantic Role Labeling (SRL), Relation Extraction (RC)



(a) Semantic Role Labeling. (b) Relation Classification.

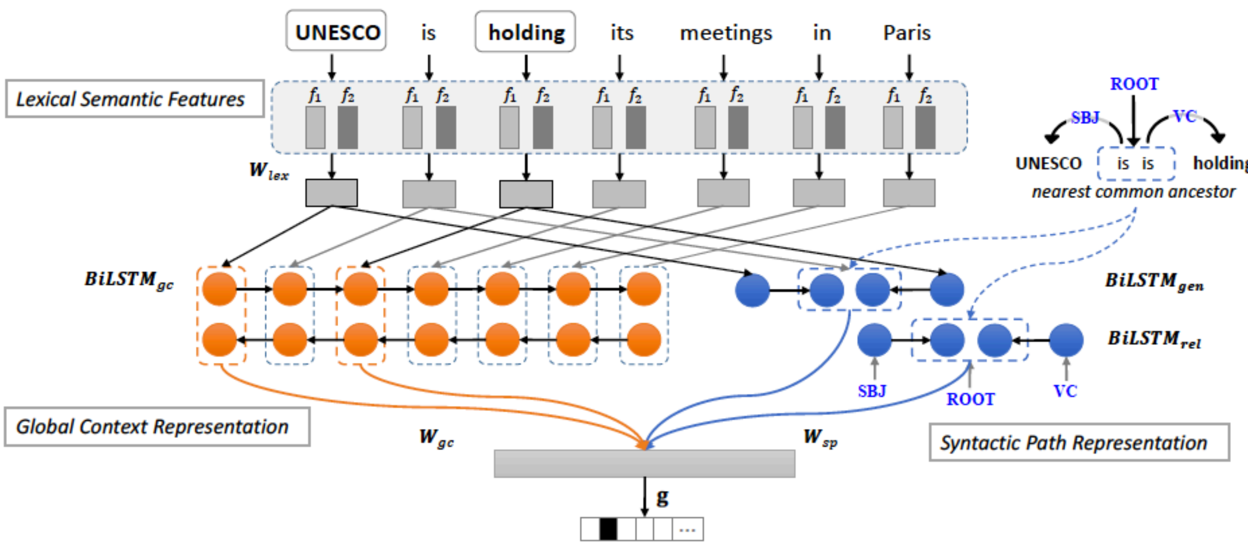
- The parsing path features are very important, but they are very sparse
  - Use LSTM to represent paths
  - All of word, POS tags and relations can be inputted



Michael Roth and Mirella Lapata. Neural Semantic Role Labeling with Dependency Path Embeddings. ACL 2016.

As Input Features

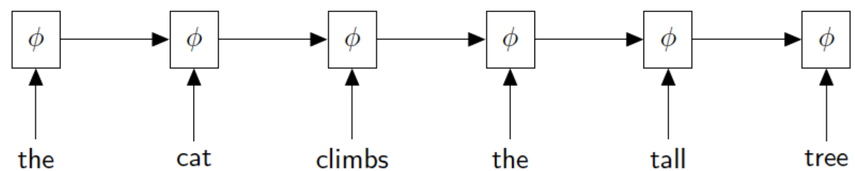
- Joint learning of SRL and RC
  - Multi-task learning



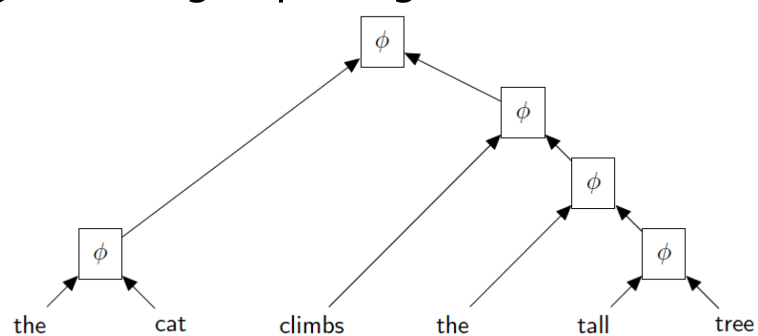
Jiang Guo, Wanxiang Che, Haifeng Wang and Ting Liu. A Unified Architecture for Semantic Role Labeling and Relation Classification. Coling 2016.

As Input Structures

- Recurrent Neural Networks
  - Composing sequentially

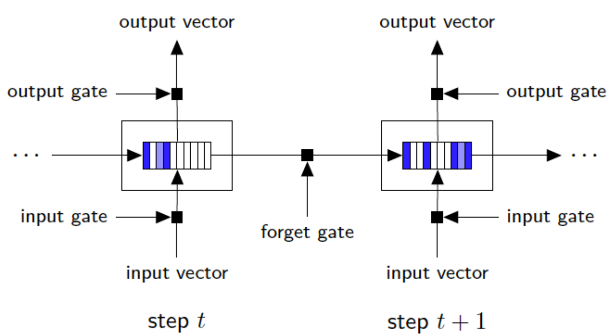


- Recursive Neural Networks
  - Use parse trees as input structures
  - Composing according to parsing structures

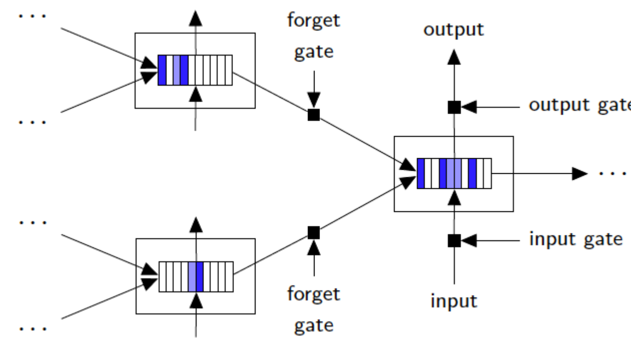


As Input Structures

- Standard LSTM

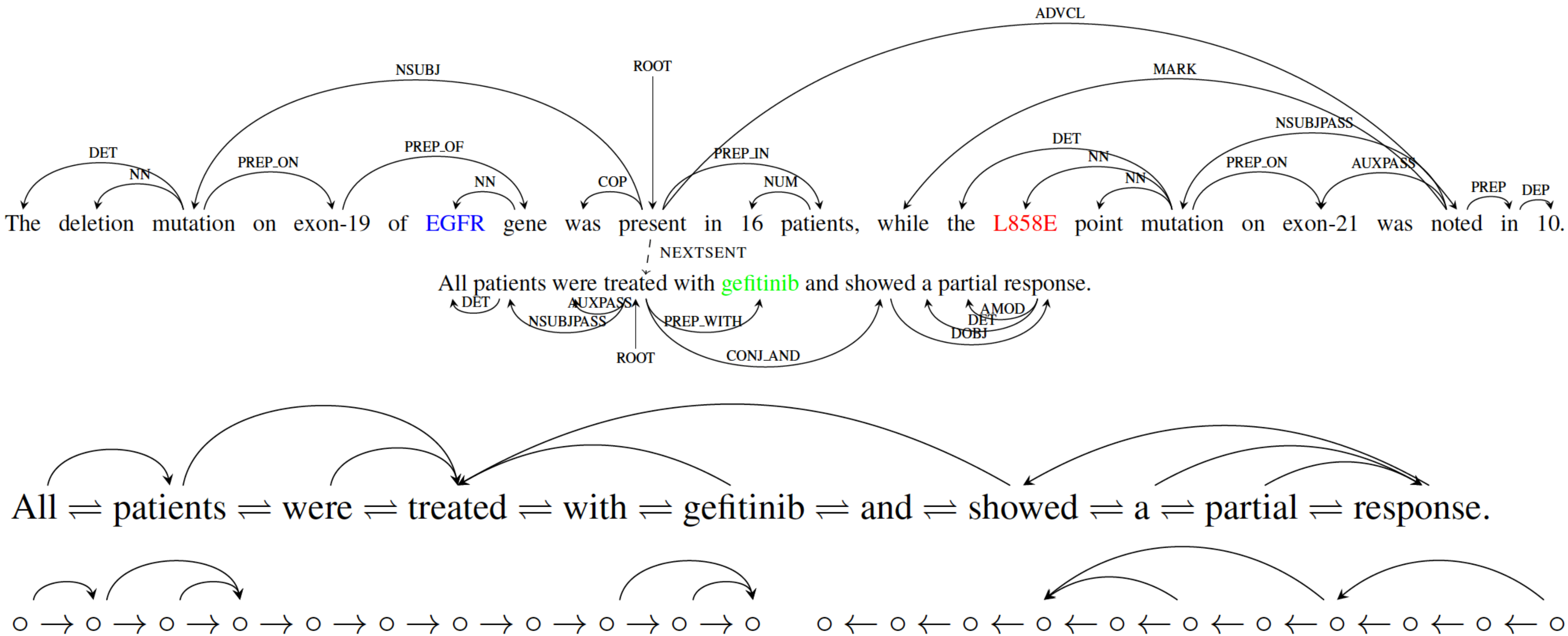


- Tree-LSTM



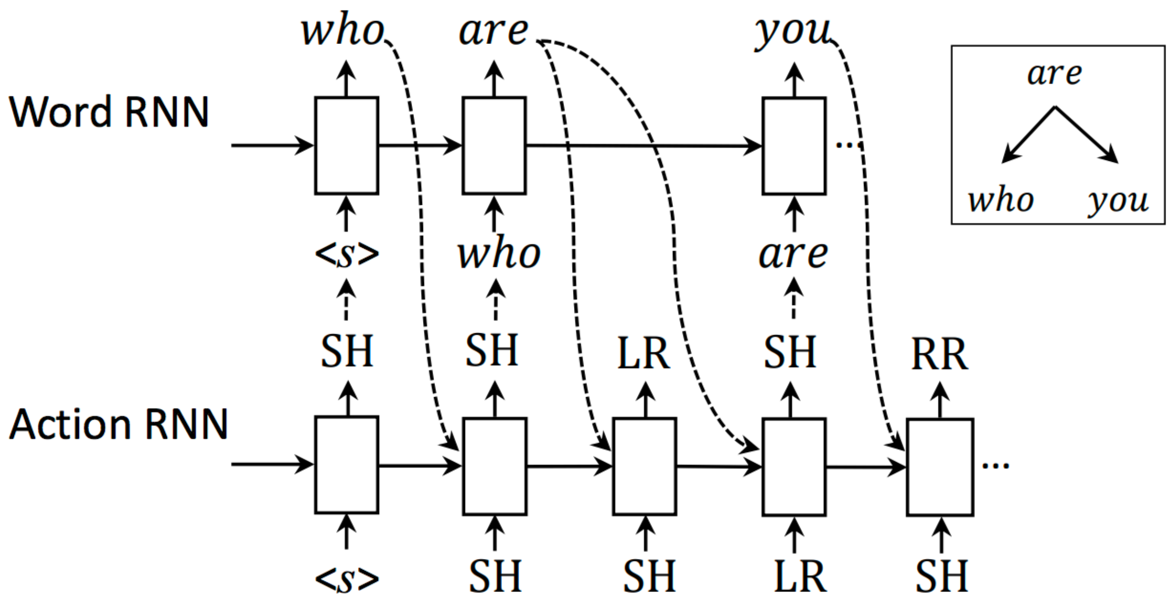
- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. ACL 2015.
- Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. ICML 2015.

As Input Structures



Peng, N., Poon, H., Quirk, C., Toutanova, K., & Yih, W. 2017 Apr 5. Cross-Sentence N-ary Relation Extraction with **Graph LSTMs**. Transactions of the Association for Computational Linguistics.

As Input Structures

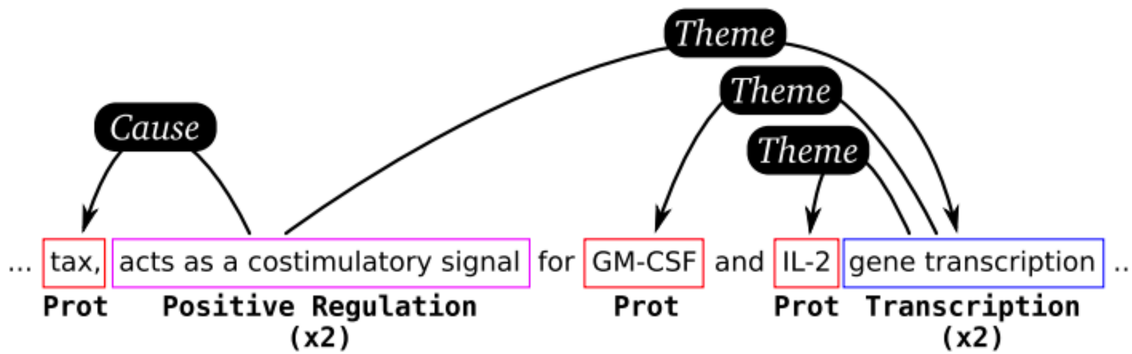


Dependency Decoder

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li and Ming Zhou. Sequence-to-Dependency **Neural Machine Translation**. ACL 2017.

As Structured Prediction

- Event Extraction as Dependency Parsing



David McClosky, Mihai Surdeanu, and Christopher D. Manning. Event Extraction as Dependency Parsing. ACL 2011.

As Structured Prediction

- Disfluency detection for speech recognition

$$\text{I want a flight [} \underbrace{\textit{to Boston}}_{\text{RM}} + \underbrace{\{\textit{um}\}}_{\text{IM}} \ \underbrace{\text{to Denver}}_{\text{RP}} \text{]}$$

- Transition System <*O*, *S*, *B*, *A*>
  - *output* (*O*) : represent the words that have been labeled as fluent
  - *stack* (*S*) : represent the partially constructed disfluency chunk
  - *buffer* (*B*) : represent the sentences that have not yet been processed
  - *action* (*A*) : represent the complete history of actions taken by the transition system
    - OUT: which moves the first word in the *buffer* to the output and clears out the *stack* if it is not empty
    - DEL: which moves the first word in the *buffer* to the *stack*

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang and Ting Liu. Transition-Based Disfluency Detection using LSTMs. EMNLP 2017.

As Structured Prediction

- An Example of transition-based disfluency detection

Step	Action	Output	Stack	Buffer
0		[]	[]	[a, flight, to, boston, to, denver]
1	OUT	[a]	[]	[flight, to, boston, to, denver]
2	OUT	[a, flight]	[]	[to, boston, to, denver]
3	DEL	[a, flight]	[to]	[boston, to, denver]
4	DEL	[a, flight]	[to, boston]	[to, denver]
5	OUT	[a, flight, to]	[]	[denver]
6	OUT	[a, flight, to, denver]	[]	[]

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang and Ting Liu. Transition-Based Disfluency Detection using LSTMs. EMNLP 2017.

Conclusion

- As Information Extraction Rules
- As Input Features
- As Input Structures
- As Structured Prediction

Course Summarization

- Lexical, Syntactic and Semantic Analysis
  - Structured Prediction (Segmentation, Tagging and Parsing)
- Traditional Methods
  - Graph-based
  - Transition-based
- Neural Network Methods
  - Neural Graph-based
  - Neural Transition-based
- Applications