Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che (HIT)

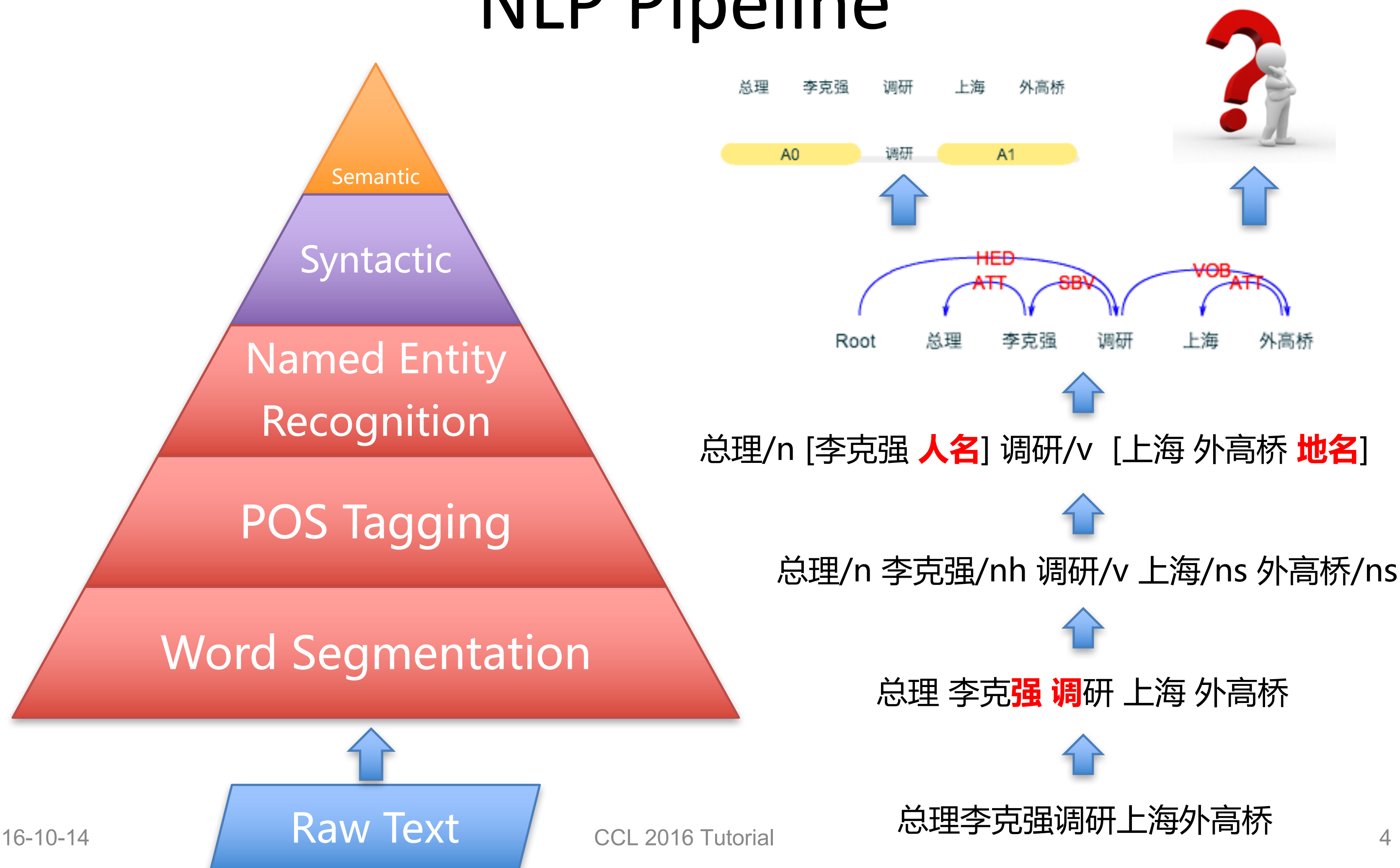
Yue Zhang (SUTD)

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

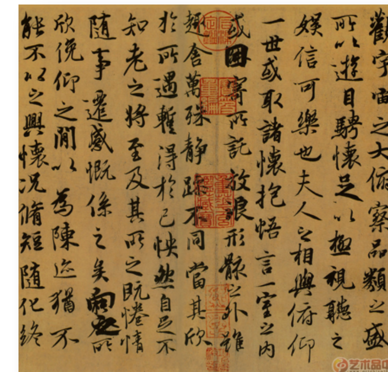
Part 1: Introduction to Lexical, Syntactic and Semantic Analysis

NLP Pipeline



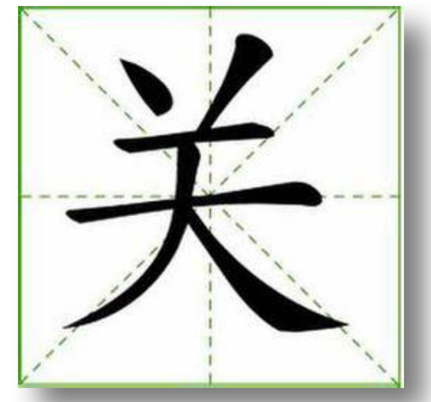
Part 1.1: Background

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

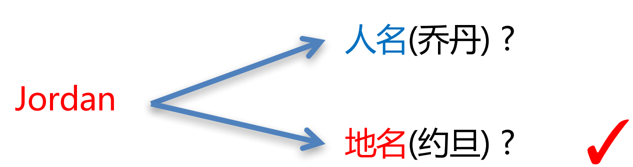


- A POS is a category of words which have similar grammatical properties
  - E.g. noun, verb, adjective
- POS tagging
  - Marking up a word in a text as a particular POS
  - based on both its definition and its context
- Ambiguities in POS Tagging
  - Time flies like an arrow.
  - 制服了敌人 vs. 穿着制服

Named Entity Recognition (NER)

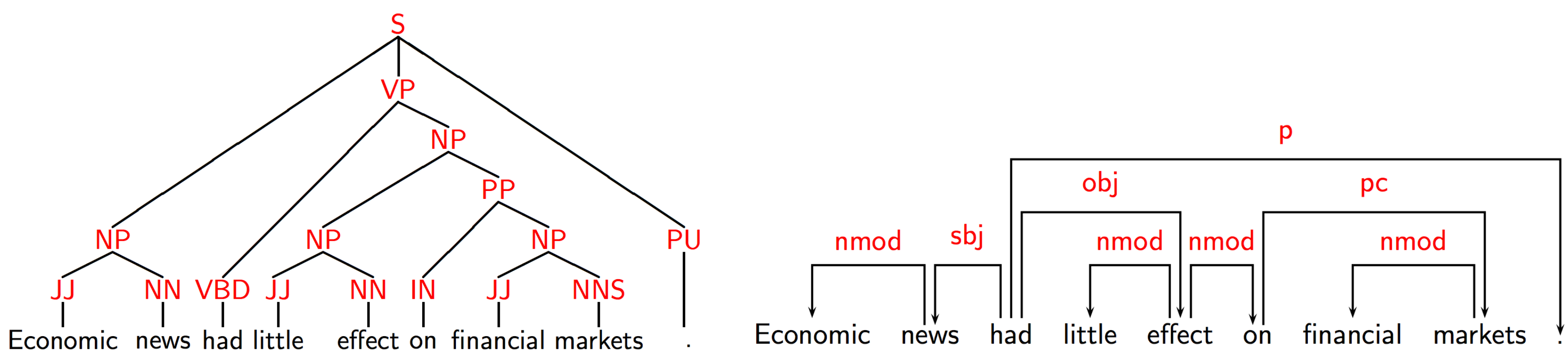
- Named Entities
  - Persons, locations, organizations, expressions of times, quantities, monetary values, percentages, etc
- Locating and classifying named entities in text into pre-defined categories
- Ambiguities in NER

Kerry to visit Jordan, Israel Palestinian peace on agenda.



Syntactic Parsing

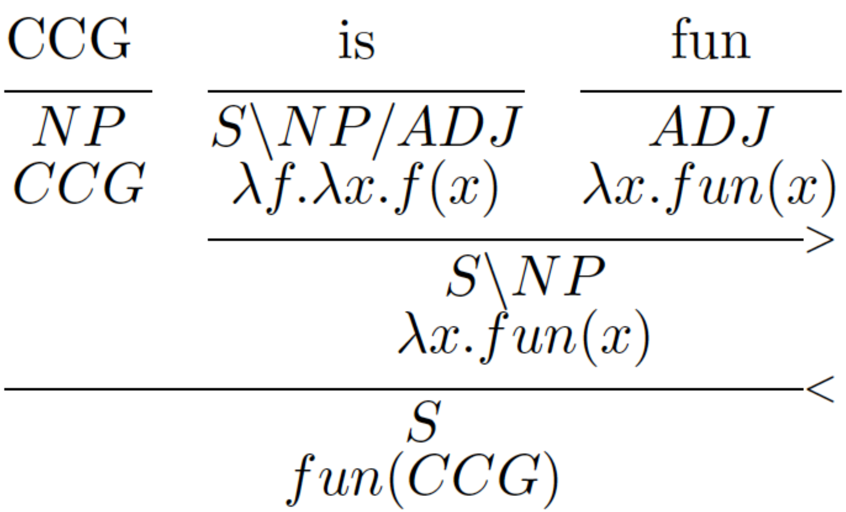
- Analyzing a natural language string conforming to the rules of a formal grammar, emphasizing subject, predicate, object, etc.
  - Constituency and Dependency Parsing



Semantic Role Labeling

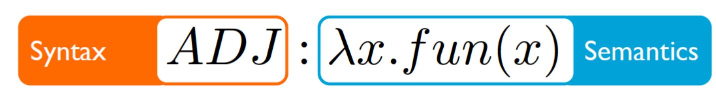
- Recognizing predicates and corresponding arguments
  - Yesterday $_{time}$, Mary $_{buyer}$ bought a shirt $_{bought\ thing}$ from Tom $_{seller}$
  - Whom $_{buyer}$ did Tom $_{seller}$ sell a shirt $_{bought\ thing}$ to, yesterday $_{time}$
- Answer "Who did what to whom when and where"
  - Question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

Combinatory Categorial Grammars (CCG)



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

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



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

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

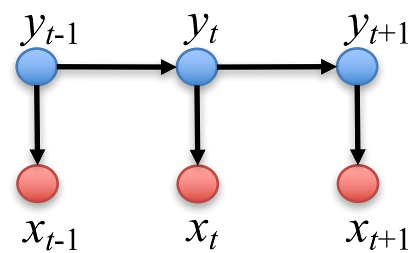
A	*bitter*	*conflict*	*with*	*global*	*implications*
NP/N	N/N	N	$(NP\backslash NP)/NP$	N/N	N

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%)

Sequence Labeling Models

HMM

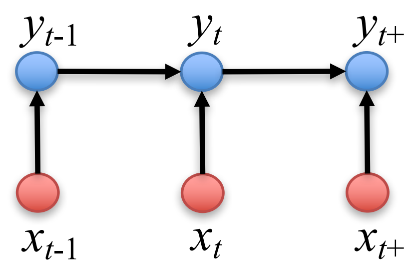
$$P(y_{[1:n]}, x_{[1:n]}) \propto \prod_{t=1}^{n} P(y_t|y_{t-1})P(x_t|y_t)$$



MEMM

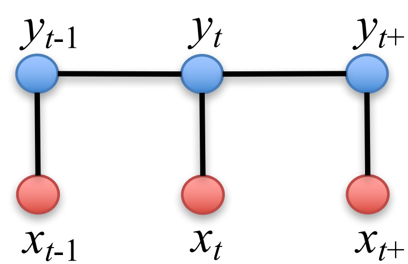
$$P(y_{[1:n]}|x_{[1:n]}) \propto \prod_{t=1}^{n} P(y_t|y_{t-1}, x_t)$$

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



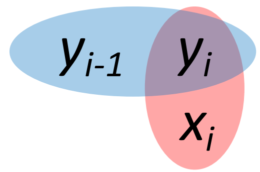
CRF

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



Features of POS Tagging with CRF

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



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

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

CRF Decoding

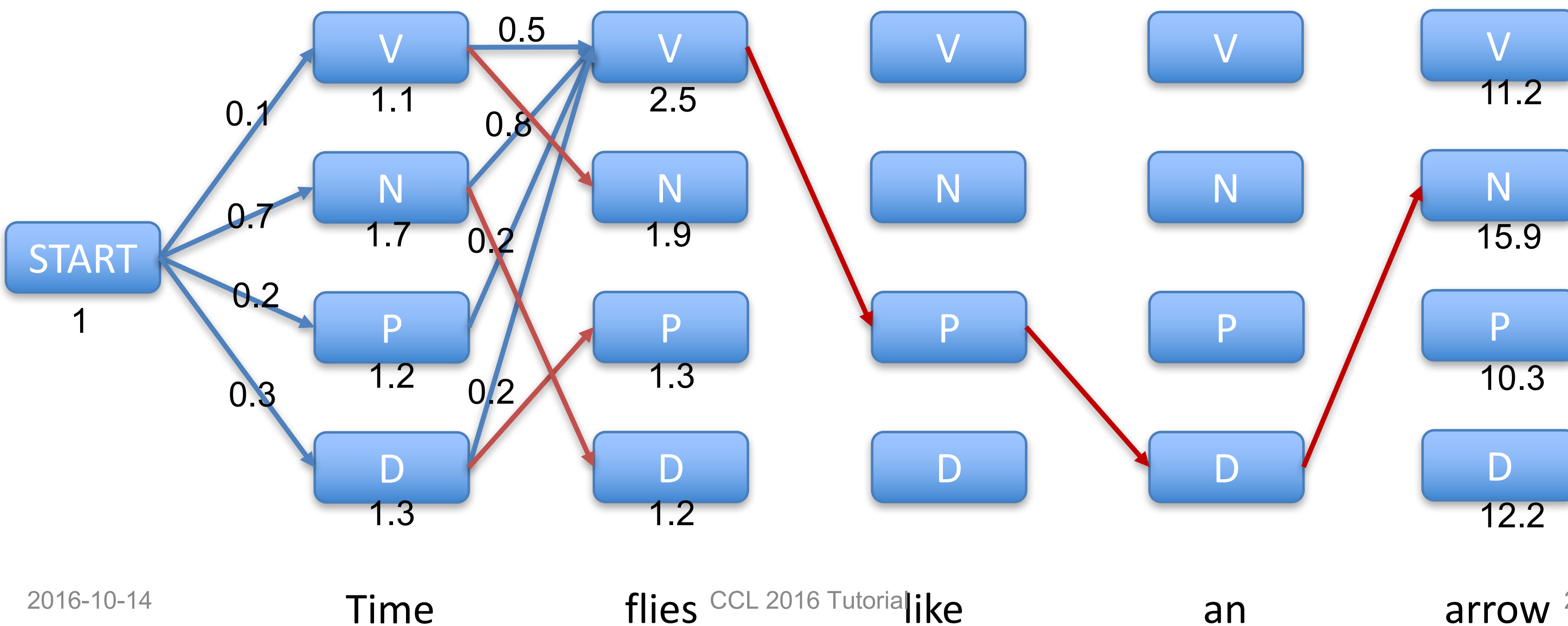
$$\underset{y_{[1:n]} \in \mathrm{GEN}(x_{[1:n]})}{\arg\max} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(x_{[1:n]}, y_i, y_{i-1})$$

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

- Dynamic Programming Algorithm
  - Viterbi Algorithm

Viterbi Algorithm

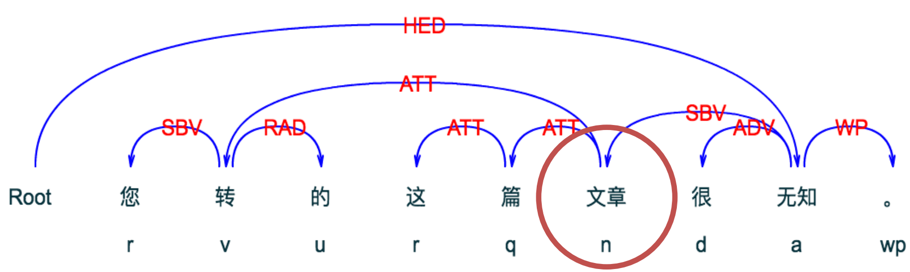
- Define a dynamic programming table
  - $\pi(i, y)$ = maximum score of a tag sequence ending in tag *y* at position *i*
- Recursive definition: $\pi(i, y) = \max_t \left(\pi(i-1, t) + \mathbf{w} \cdot \mathbf{f}(x_{[1:n]}, y, t)\right)$

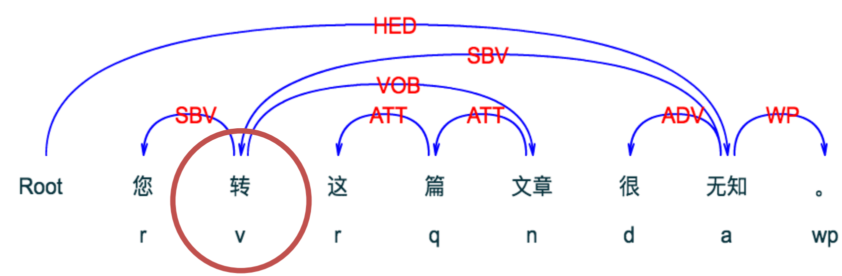


Part 1.3: Parsing

Dependency Parsing

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

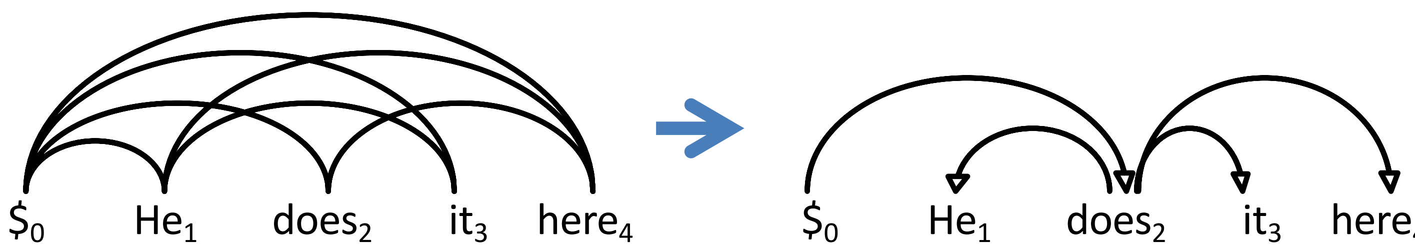




http://www.ltp-cloud.com/demo/

Graph-based Dependency Parsing

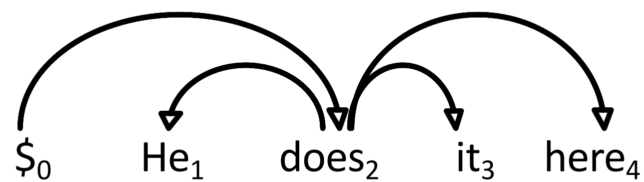
- Find the highest scoring tree from a complete dependency graph



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

First-order as an Example

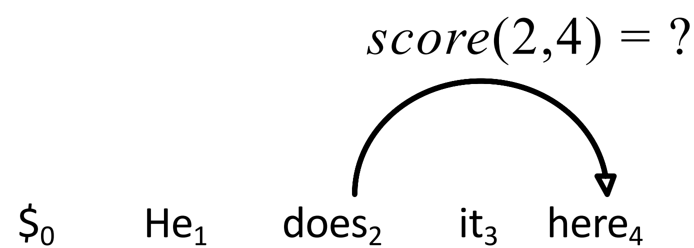
- The first-order graph-based method assumes that arcs in a tree are independent from each other (arc-factorization)
- Maximum Spanning Tree (MST) Algorithm



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

How to Score an Arc

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



- 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

- 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 $a_0 \ldots a_{i-1}$

Y → 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.

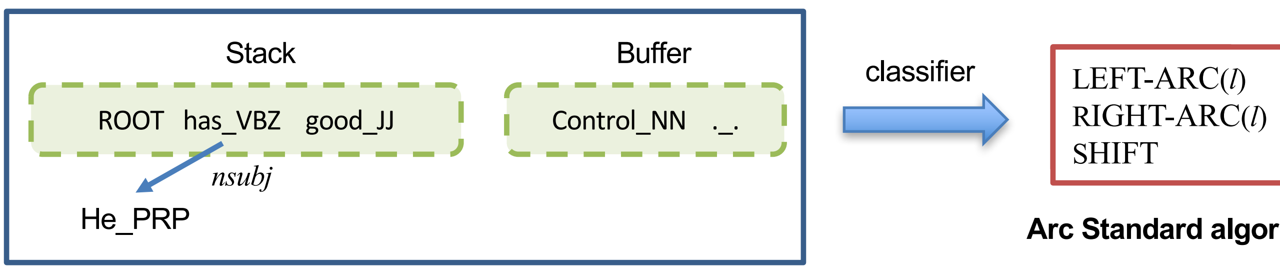
$$Y^* = \underset{Y \in \Phi(X)}{\arg\max}\, score(X, Y)$$

$$= \underset{a_0 \ldots a_m \to Y}{\arg\max} \sum_{i=0}^{m} score(X, h_i, a_i)$$

Transition-based Dependency Parsing

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

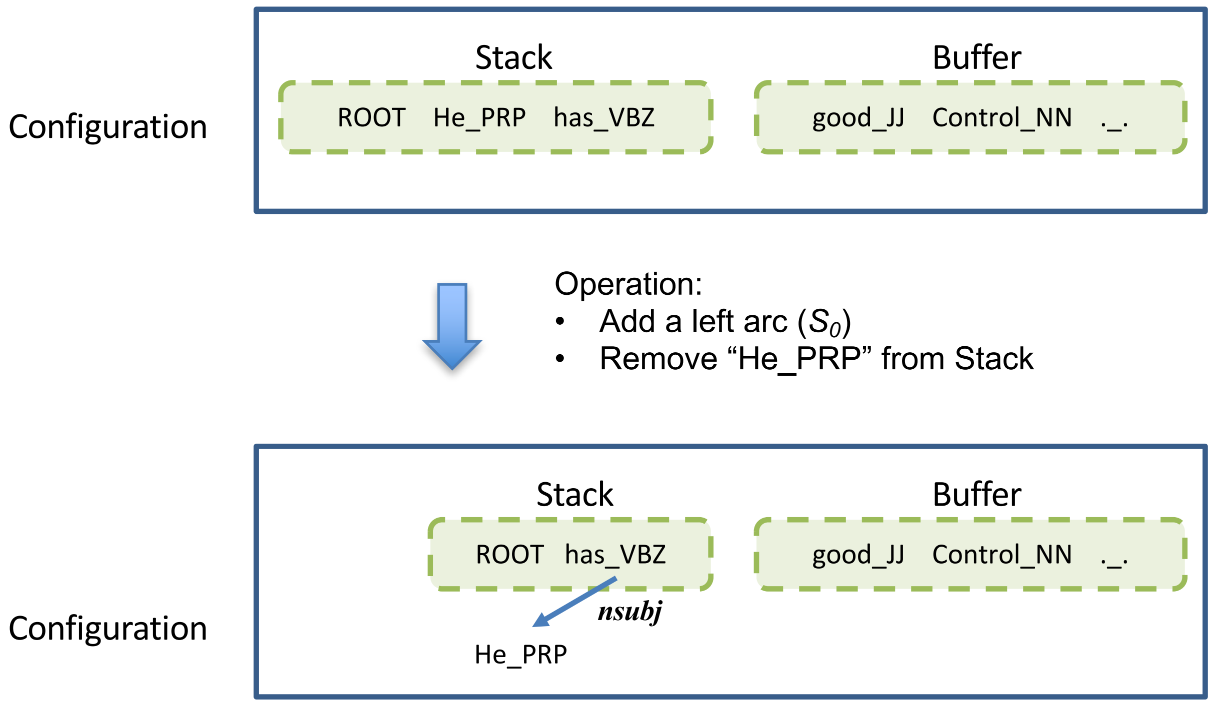
  = Stack + Buffer + Dependency Arcs



Configuration

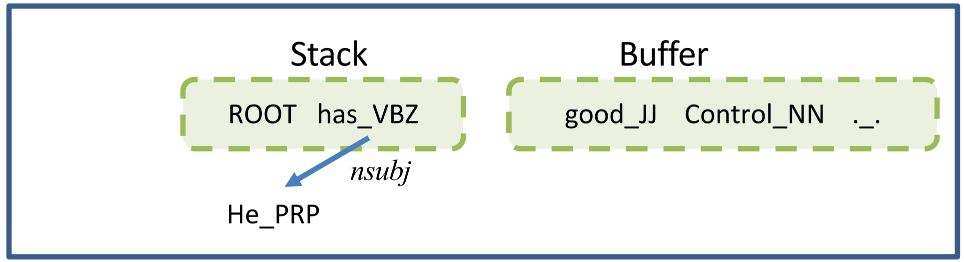
Arc Standard algorithm

Transition Action: LEFT-ARC (*l*)



Transition Action: SHIFT

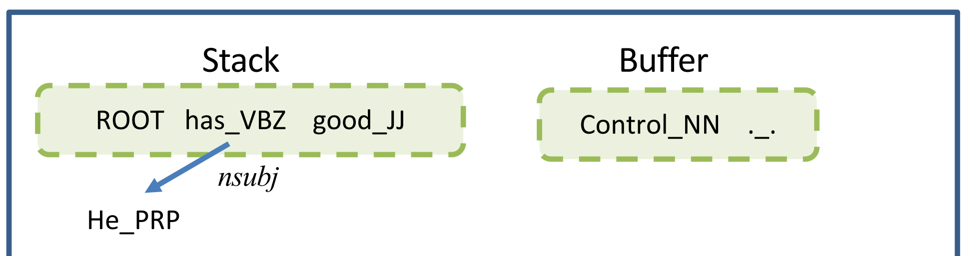
Configuration



Operation:

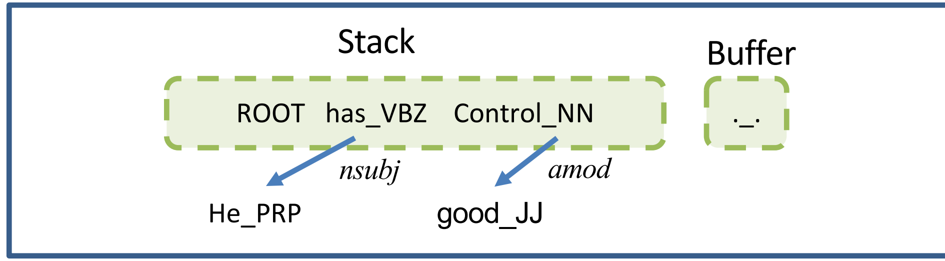
- Shift “good_JJ” from Buffer to top of Stack

Configuration



Transition Action: RIGHT-ARC (*l*)

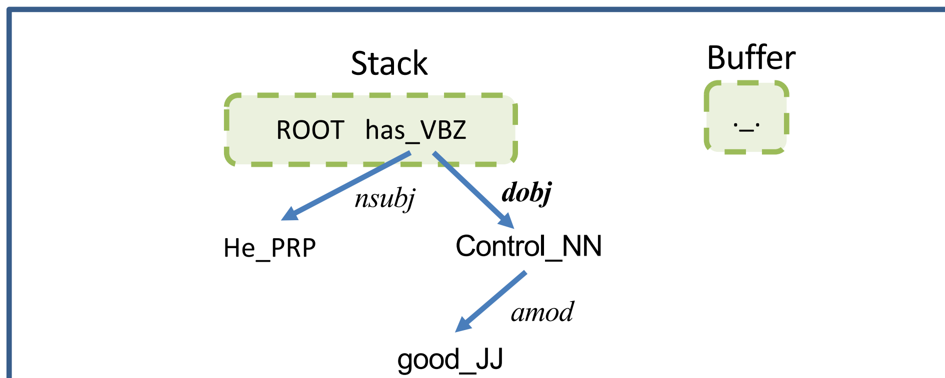
Configuration



Operation:

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

Configuration



An Example

Arc-standard Algrithm

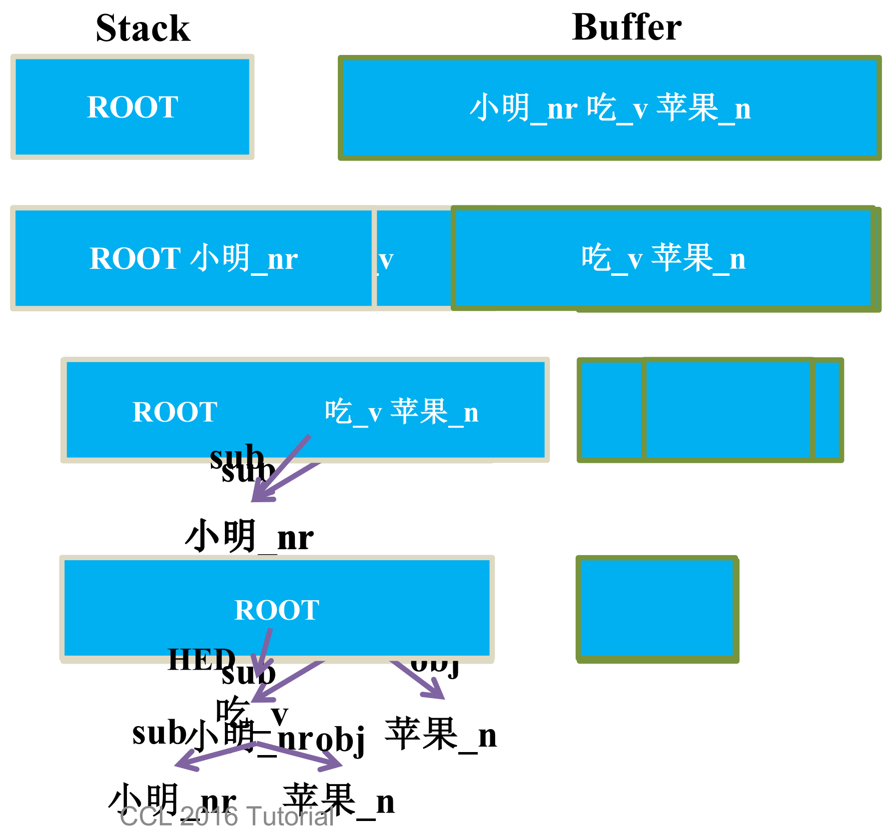
初始状态
Stack只有根节点，待处理词在Buffer中

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

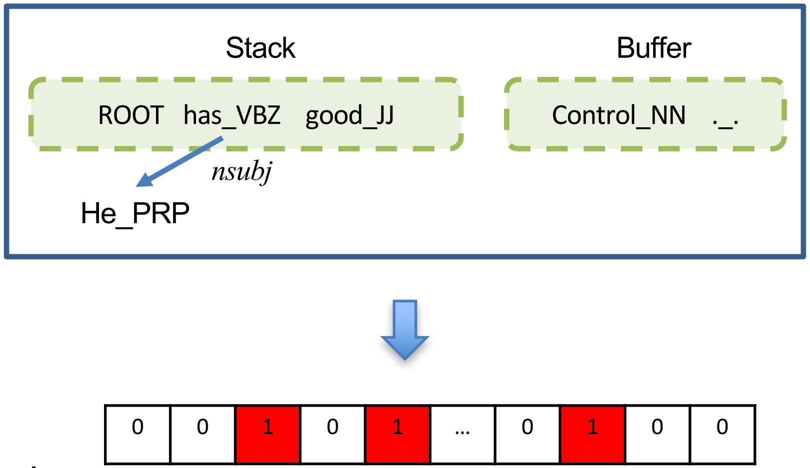
RIGHT-ARC
弹出栈顶词，生成一条弧从栈顶第二个词指向栈顶词

终结状态
Stack只有根节点，Buffer为空



Traditional Features

Configuration



Feature Vector:
- Binary
- Sparse
- High-dimensional

Feature templates: a combination of elements from the configuration.
- For example: (Zhang and Nivre, 2011): 72 feature templates

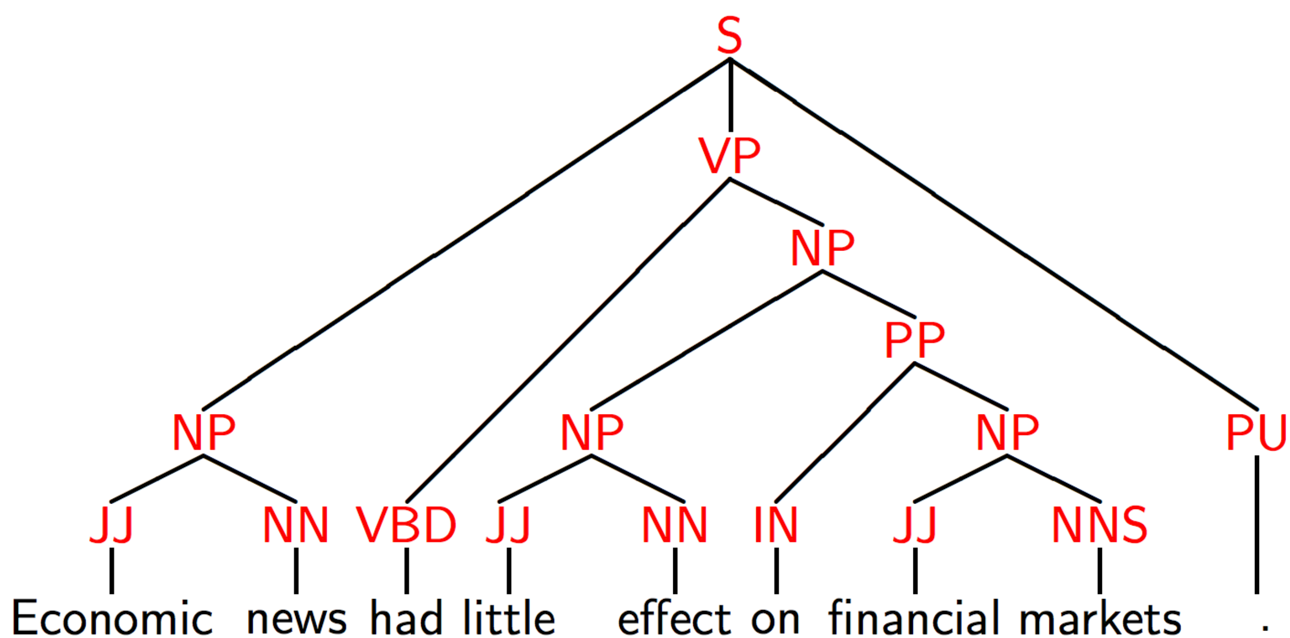
from single words
S_0wp; S_0w; S_0p; N_0wp; N_0w; N_0p; N_1wp; N_1w; N_1p; N_2wp; N_2w; N_2p;
from word pairs
S_0wpN_0wp; S_0wpN_0w; S_0wN_0wp; S_0wpN_0p; S_0pN_0wp; S_0wN_0w; S_0pN_0p N_0pN_1p
from three words
$N_0pN_1pN_2p$; $S_0pN_0pN_1p$; $S_{0h}pS_0pN_0p$; $S_0pS_{0l}pN_0p$; $S_0pS_{0r}pN_0p$; $S_0pN_0pN_{0l}p$

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;

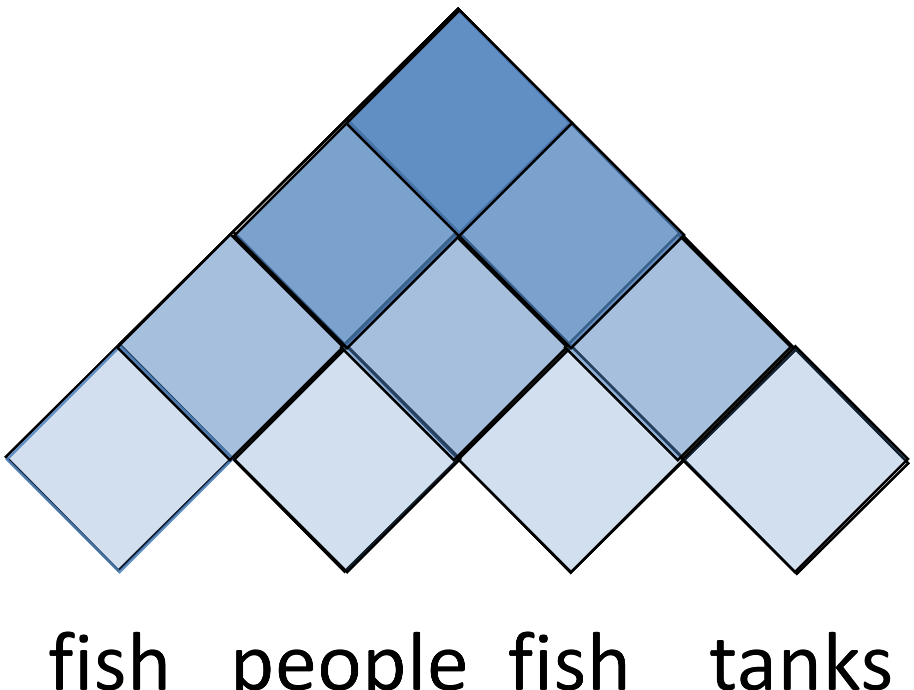
Table 2: New feature templates.
w – word; p – POS-tag; v_l, v_r – valency; l – dependency label, s_l, s_r – labelset.

Constituency Parsing



Constituency Parsing

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



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 $s = x_1 \ldots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.
Initialization:
For all $i \in \{1 \ldots n\}$, 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 $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$
    - Set $j = i + l$
    - For all $X \in N$, calculate

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

and

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

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

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!