Incremental Structured Prediction Using a Global Learning and Beam-Search Framework

Yue Zhang[1], Meishan Zhang[2], Ting Liu[2]

Singapore University of Technology and Design[1]
yue_zhang@sutd.edu.sg
Harbin Institute of Technology, China[2]
{mszhang, tliu}@ir.hit.edu.cn

Outline

Introduction	**Applications**
Analysis	**ZPar**

Outline

Introduction	**Applications**
Analysis	**ZPar**

Introduction

- Structured prediction problems
- An overview of the transition system
- Algorithms in details
  - Beam-search decoding
  - Online learning using averaged perceptron

Introduction

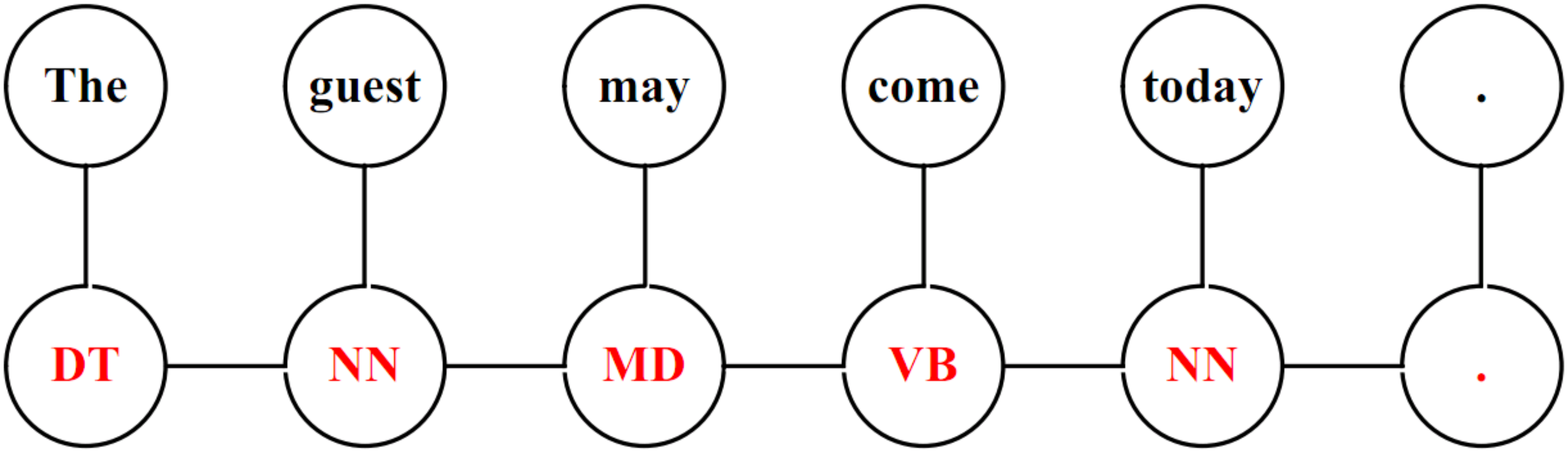
- **Structured prediction problems**
- An overview of the transition system
- Algorithms in details
  - Beam-search decoding
  - Online learning using averaged perceptron

Structured prediction problems

- Two important tasks in NLP
  - Classification
    - Output is a single label
    - Examples
      - Document classification
      - Sentiment analysis
      - Spam filtering
  - Structured prediction
    - Output is a set of *inter-related* labels or a structure

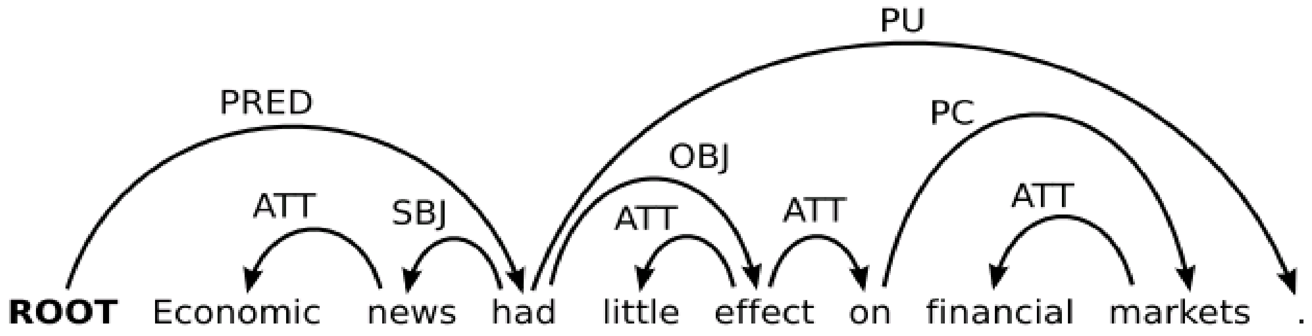
Structured prediction problems

- POS Tagging



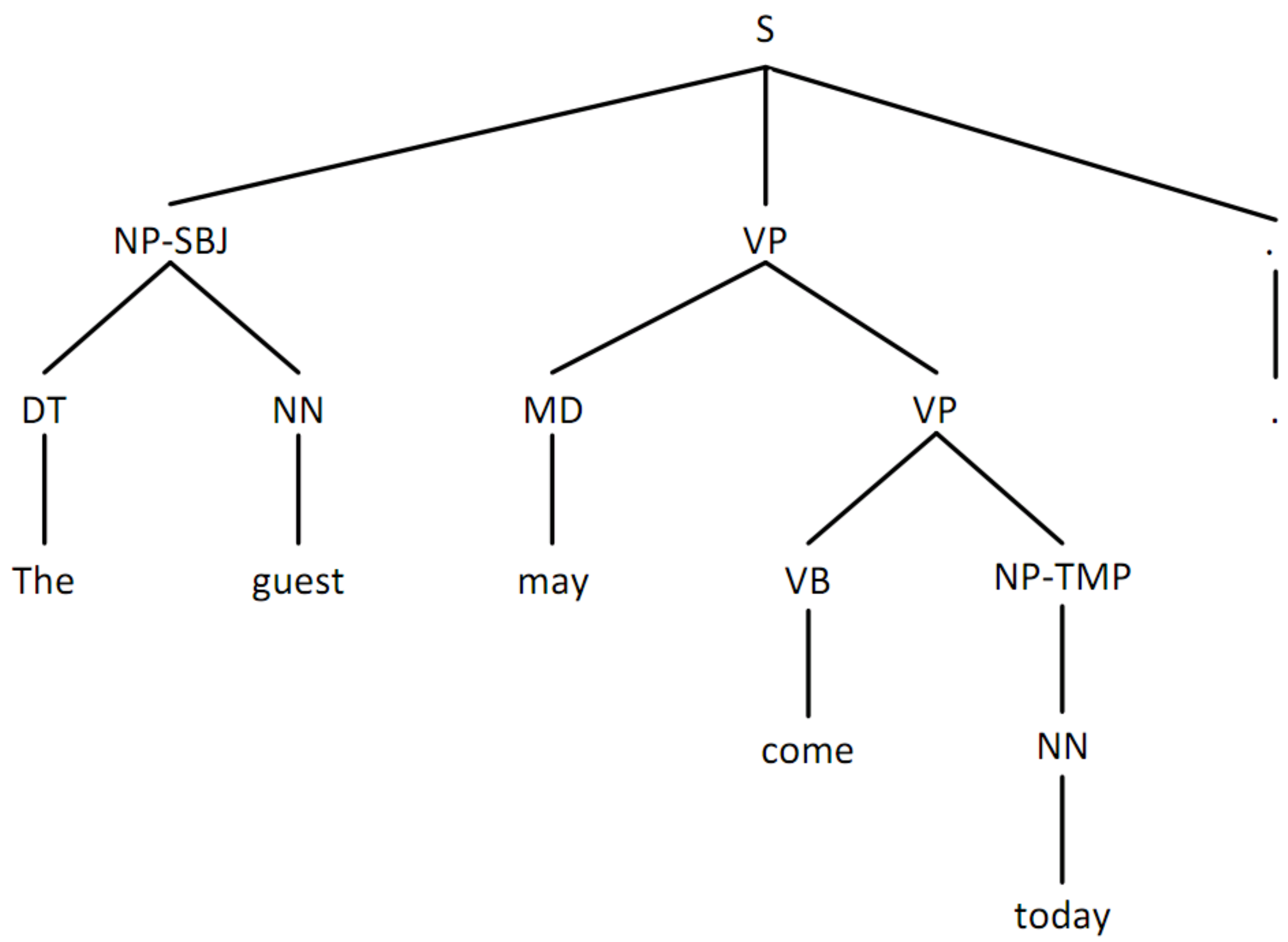
Structured prediction problems

- Dependency parsing



Structured prediction problems

- Constituent parsing



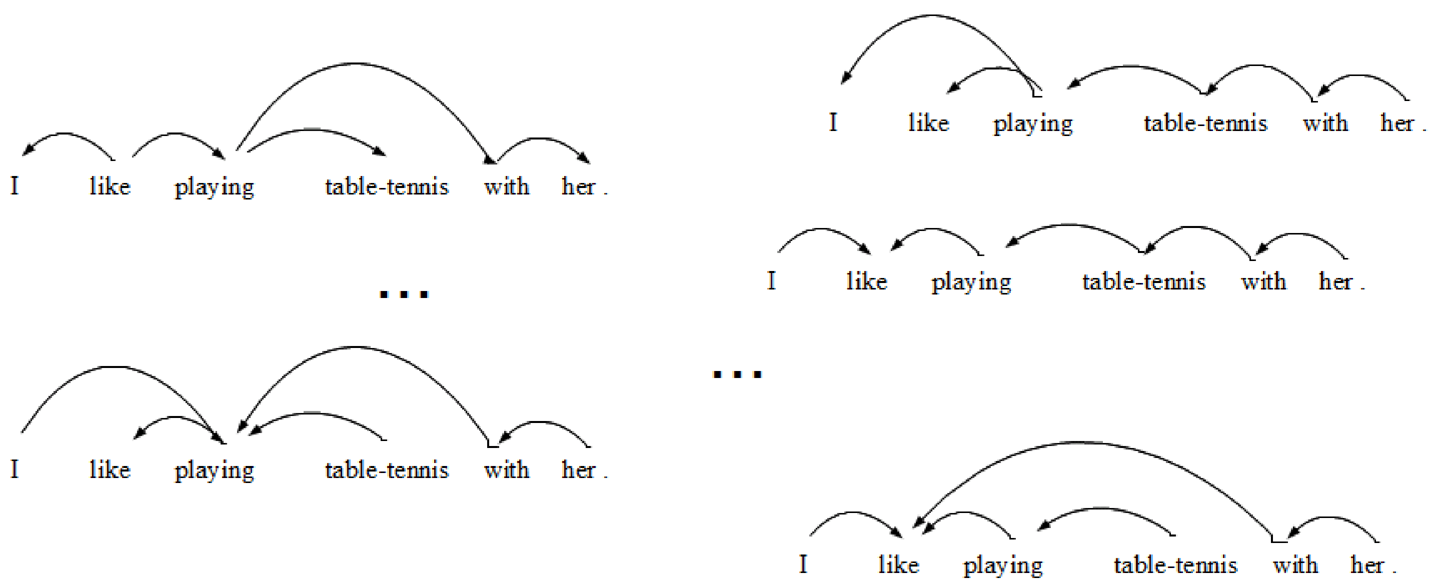
Structured prediction problems

- Machine Translation

总统 (president)	将 (will)	于 (in)	四月 (April)	来 (come)	伦敦 (London)	访问 (visit)
The	President	will	visit	London	in	April

Structured prediction problems

- Traditional solution
  - Score each candidate, select the highest-scored output
  - Search-space typically exponential



- ✓ **Over 100 possible trees for this seven-word sentence.**
- ✓ **Over one million trees for a 20-word sentence.**

Structured prediction problems

- One solution: dynamic programing methods
  - Independence assumption on features
  - Local features with global optimization
  - Solve the exponential problems in polynomial time

Structured prediction problems

- One solution: dynamic programing methods
  - Independence assumption on features
  - Local features with global optimization
  - Solve the exponential problems in polynomial time

- Examples
  - POS tagging: Markov assumption, $p(t_i|t_{i-1}\ldots t_1) = p(t_i|t_{i-1})$
    - Viterbi decoding
  - Dependency parsing: arc-factorization
    - 1st-order MST decoding

Structured prediction problems

- The learning problem
  - How to score candidate items such that a higher reflects a more correct candidate.

- Examples
  - POS-tagging: HMM, CRF
  - Dependency parsing: MIRA

Structured prediction problems

- Transition-based methods with beam search decoding
  - A framework for structured prediction

Structured prediction problems

- Transition-based methods with beam search decoding
  - A framework for structured prediction
  - Incremental state transitions
    - Use transition actions to build the output
    - Typically left to right
    - Typically linear time

Structured prediction problems

- Transition-based methods with beam search decoding
  - A framework for structured prediction
  - Incremental state transitions
  - The search problem
    - To find a highest-score action sequence out of an exponential number of sequences, rather than scoring structures directly
    - Beam-search (non-exhaustive decoding)

Structured prediction problems

- Transition-based methods with beam search decoding
  - A framework for structured prediction
  - Incremental state transitions
  - The search problem
  - Non-local features
    - Arbitrary features enabled by beam-search

Structured prediction problems

- Transition-based methods with beam search decoding
  - A framework for structured prediction
  - Incremental state transitions
  - The search problem
  - Non-local features
  - The learning problem
    - To score candidates such that a higher-scored action sequence leads to a more correct action sequence
    - Global discriminative learning

Structured prediction problems

- Transition-based methods with beam search decoding
  - A framework for structured prediction
  - Incremental state transitions
  - The search problem
  - Non-local features
  - The learning problem
- The framework of this tutorial

  (Zhang and Clark, CL 2011)

Structured prediction problems

- Transition-based methods with beam search decoding
- The framework of this tutorial
- Very high accuracies and efficiencies using this framework
  - Word segmentation (Zhang and Clark, ACL 2007)
  - POS-tagging
  - Dependency parsing (Zhang and Clark, EMNLP 2008; Huang and Sagae ACL 2010, Zhang and Nirve, ACL 2011, Zhang and Nirve, COLING 2012; Goldberg et al., ACL 2013)
  - Constituent parsing (Collins and Roark, ACL 2004; Zhang and Clark, IWPT 2009; Zhu et al. ACL 2013)
  - CCG parsing (Zhang and Clark, ACL 2011)
  - Machine translation (Liu, ACL 2013)
  - Joint word segmentation and POS-tagging (Zhang and Clark, ACL 2008; Zhang and Clark, EMNLP 2010)
  - Joint POS-tagging and dependency parsing (Hatori et al. IJCNLP 2011; Bohnet and Nirve, EMNLP 2012)
  - Joint word segmentation, POS-tagging and parsing (Hatori et al. ACL 2012; Zhang et al. ACL2013; Zhang et al. ACL2014)
  - Joint morphological analysis and syntactic parsing (Bohnet et al., TACL 2013)

Structured prediction problems

- Transition-based methods with beam search decoding
- The framework of this tutorial
- Very high accuracies and efficiencies using this framework
- General
  - Can apply to any structured predication tasks, which can be transformed into an incremental process

Introduction

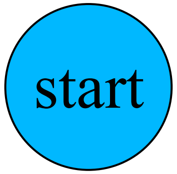
- **Structured prediction problems**
- **An overview of the transition system**
- **Algorithms in details**
  - **Beam-search decoding**
  - **Online learning using averaged perceptron**

A transition system

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

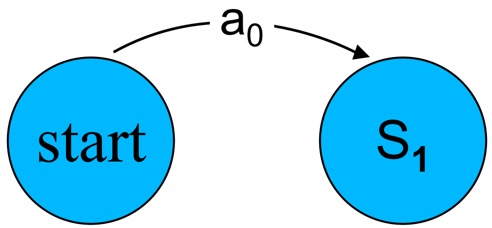
A transition system

- Automata



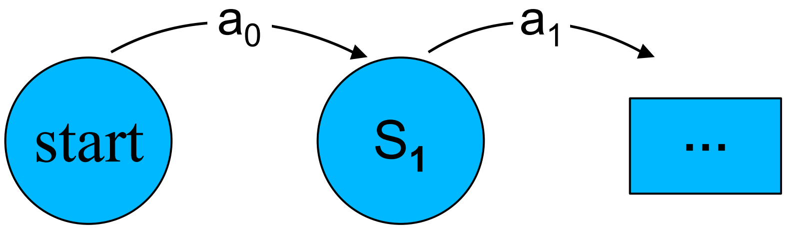
A transition system

- Automata



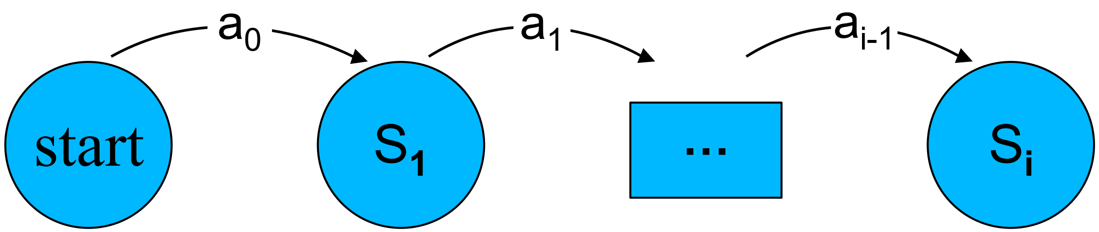
A transition system

- Automata



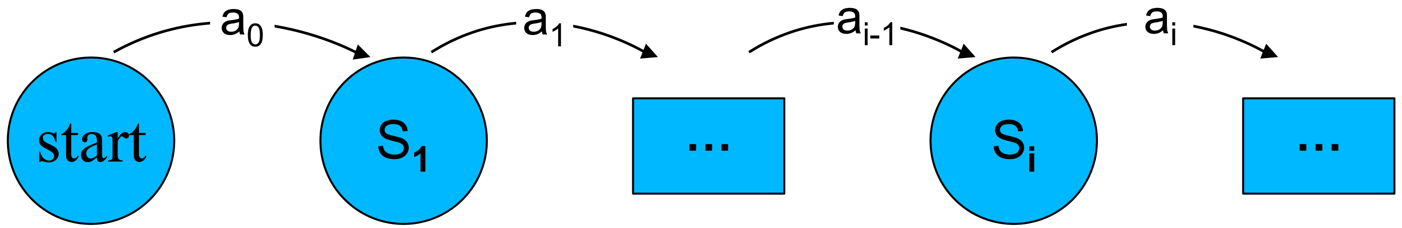
A transition system

- Automata



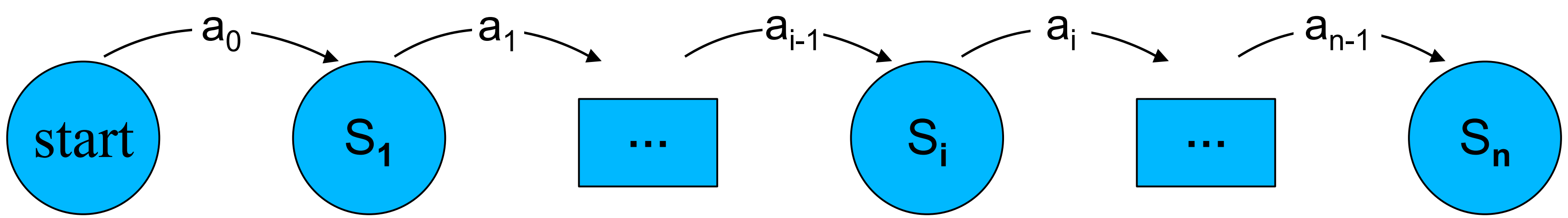
A transition system

- Automata



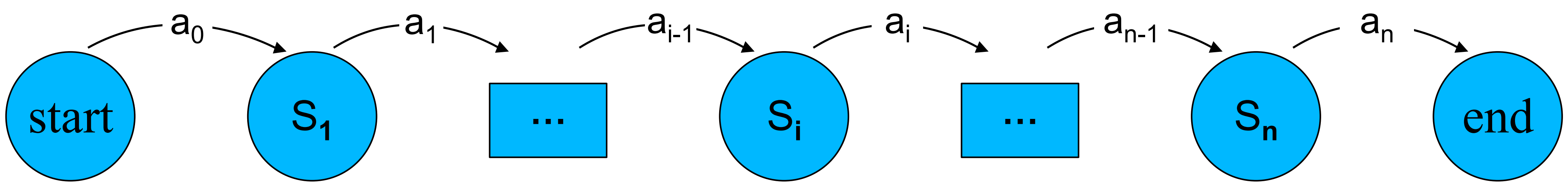
A transition system

- Automata



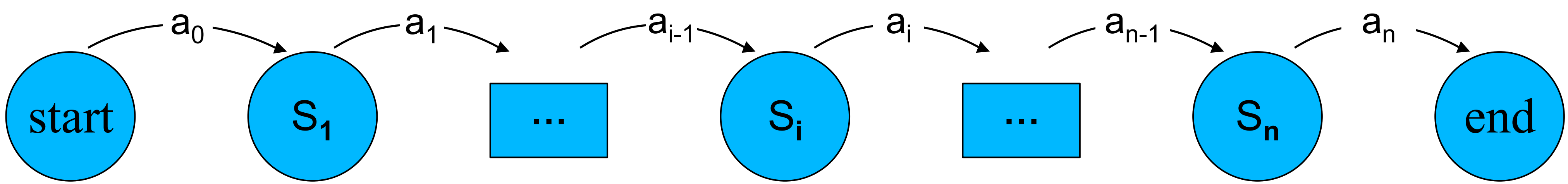
A transition system

- Automata



A transition system

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



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

A transition-based POS-tagging example

- POS tagging

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

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

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
```

Introduction

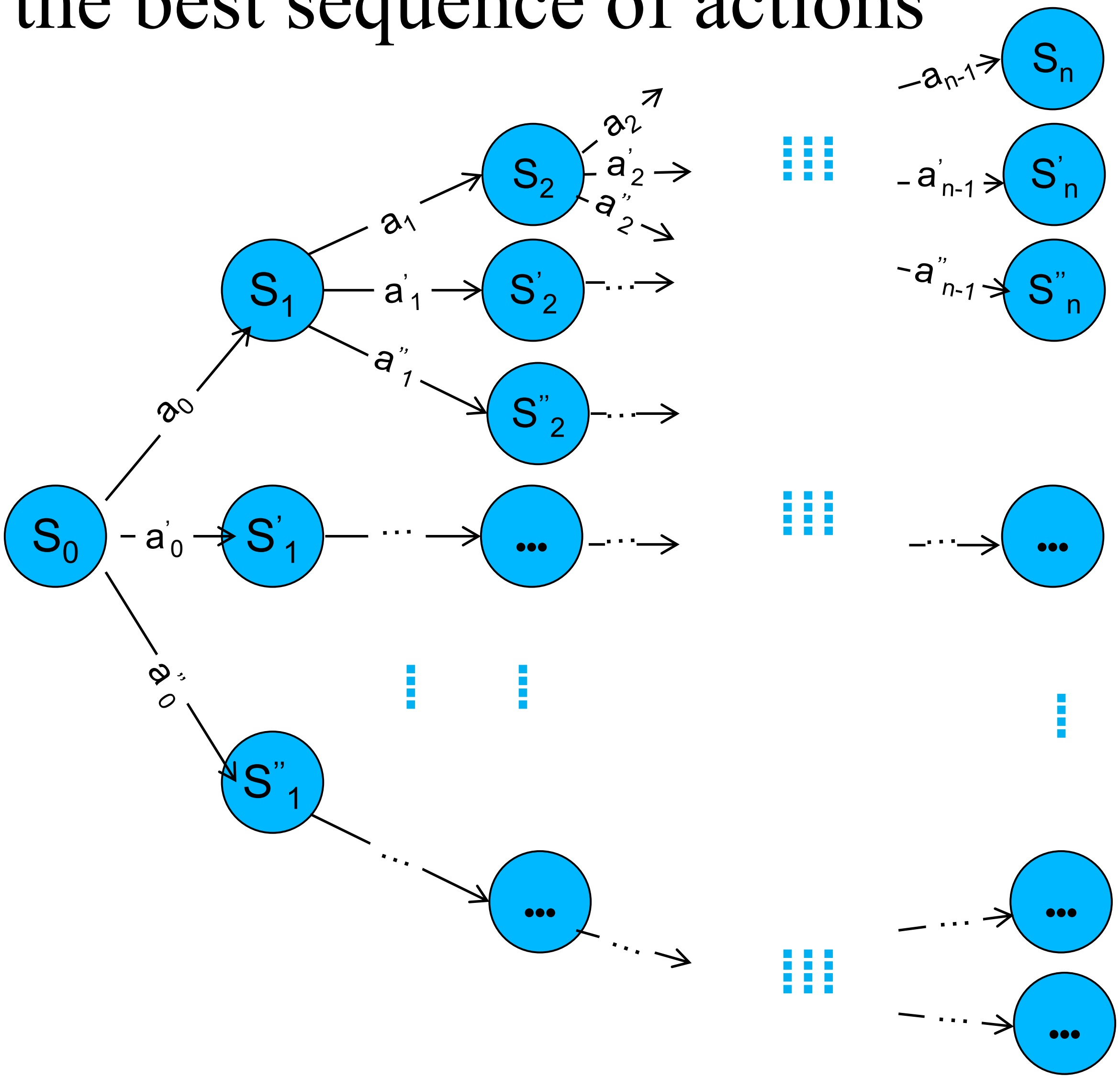
- **Structured prediction problems**
- **An overview of the transition system**
- **Algorithms in details**
  - **Beam-search decoding**
  - **Online learning using averaged perceptron**

Introduction

- Structured prediction problems
- An overview of the transition system
- Algorithms in details
  - Beam-search decoding
  - Online learning using averaged perceptron

Search

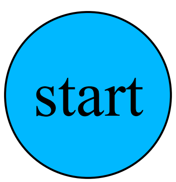
- Find the best sequence of actions



Search

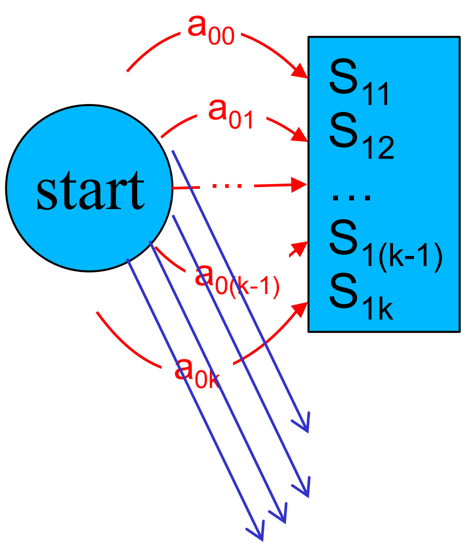
- Dynamic programming
  - Optimum sub-problems are recorded according to dynamic programming signature
  - Infeasible if features are non-local (which are typically useful)
- One solution
  - Greedy classification
    - Input: $\mathrm{S}_{\mathbf{i}}$
    - Output: $a_i = \underset{a'}{argmax}\, w \cdot f(\mathrm{S}_{\mathbf{i}}, a')$
- For better accuracies: beam-search decoding

Beam-search decoding



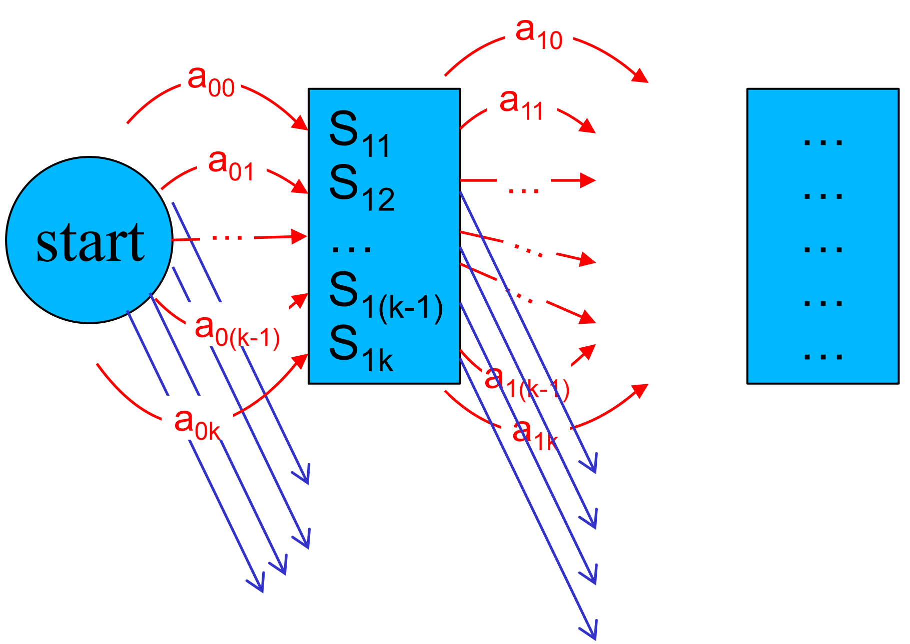
Zhang and Clark, CL 2011

Beam-search decoding



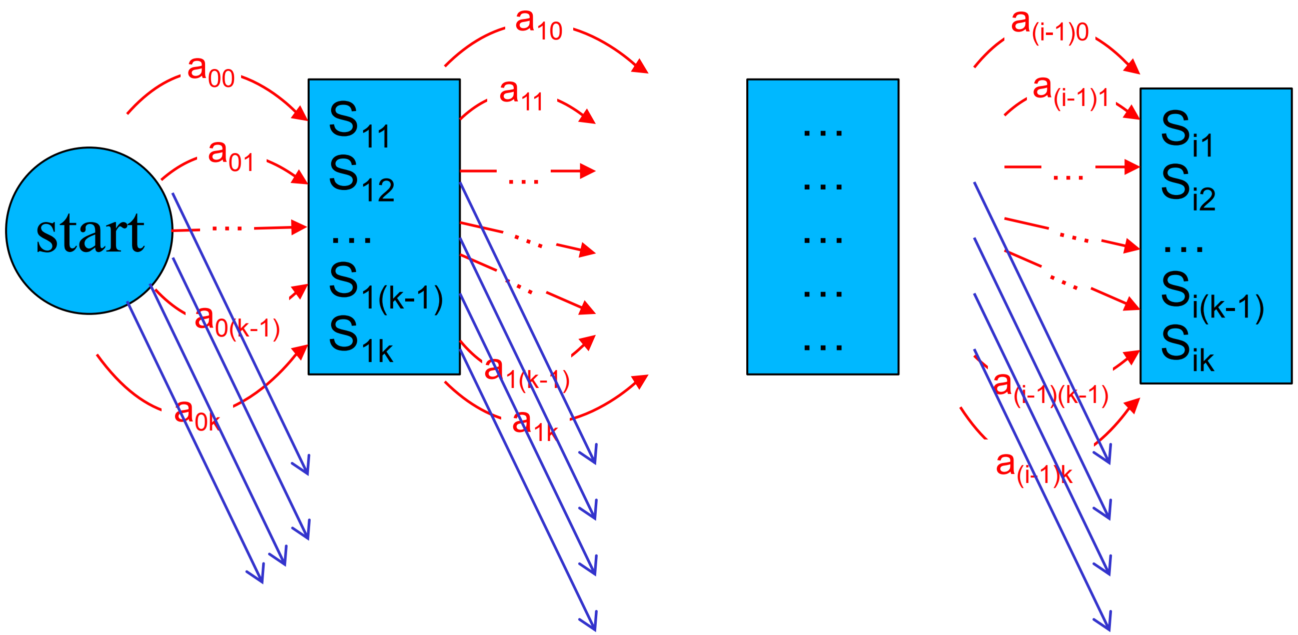
Zhang and Clark, CL 2011

Beam-search decoding



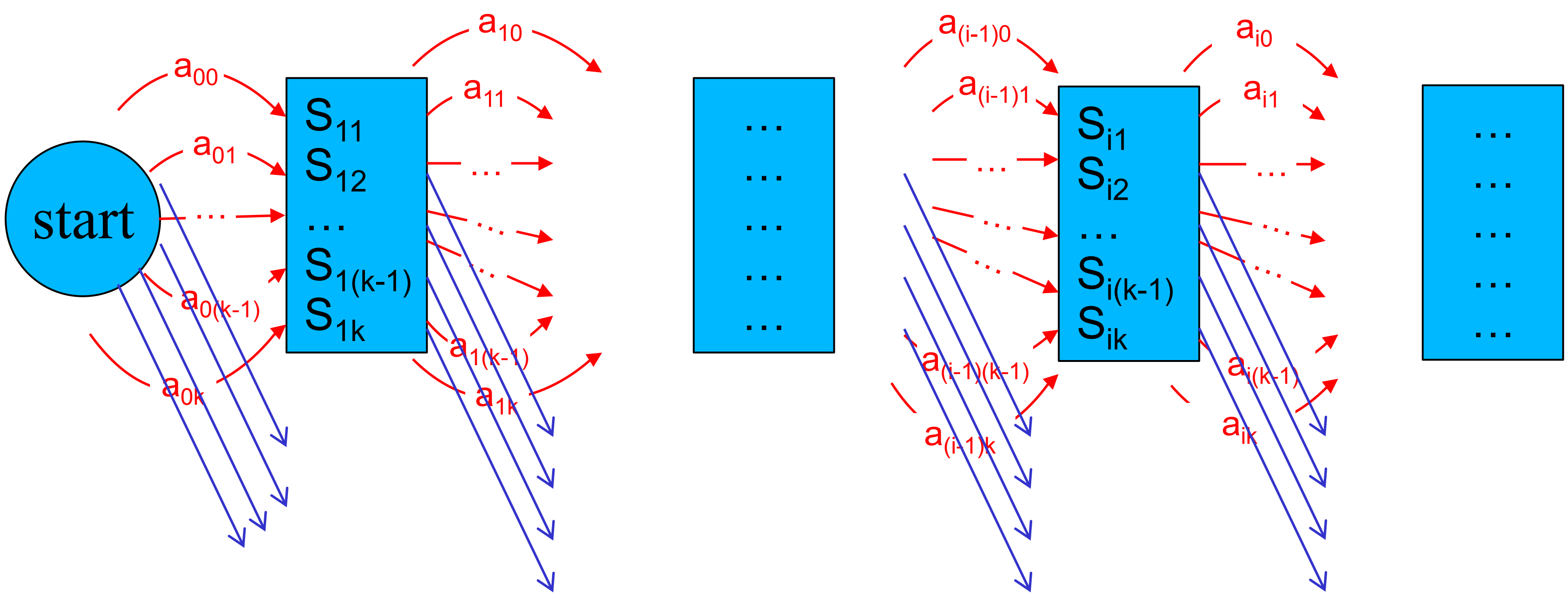
Zhang and Clark, CL 2011

Beam-search decoding



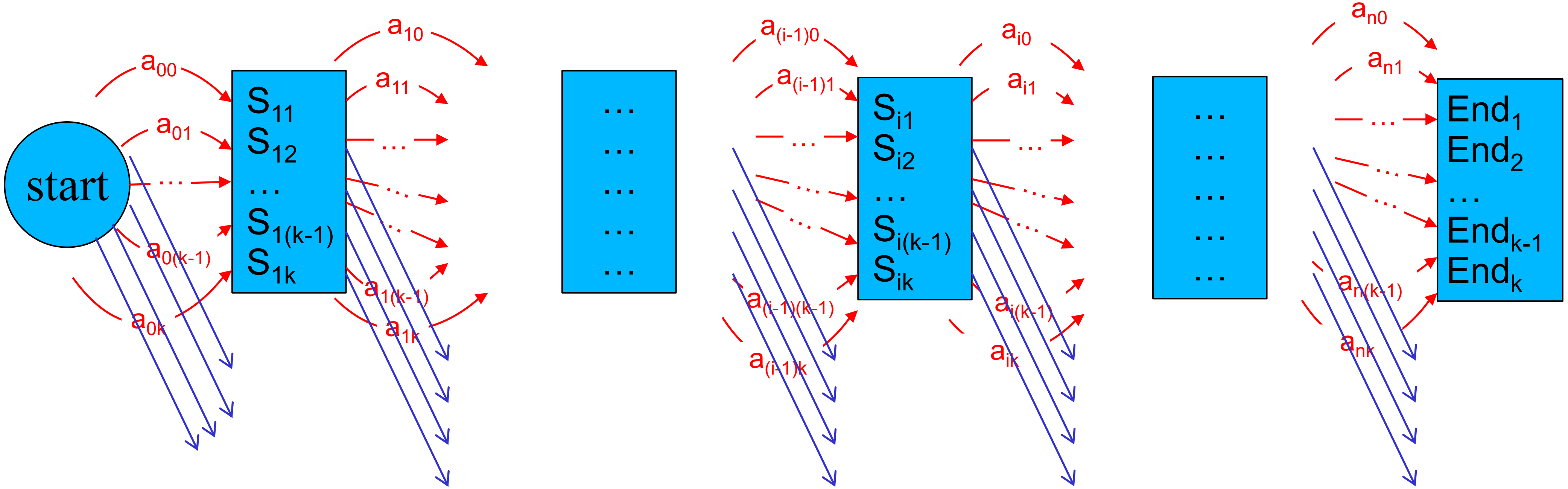
Zhang and Clark, CL 2011

Beam-search decoding



Zhang and Clark, CL 2011

Beam-search decoding



Zhang and Clark, CL 2011

Beam-search decoding

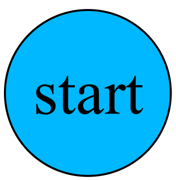
```
function BEAM-SEARCH(problem, agenda, candidates, B)

  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 and Clark, CL 2011

Beam-search decoding

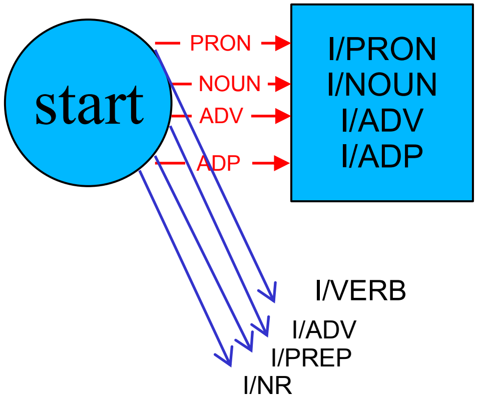
- An example: POS-tagging
  - I like reading books



Zhang and Clark, CL 2011

Beam-search decoding

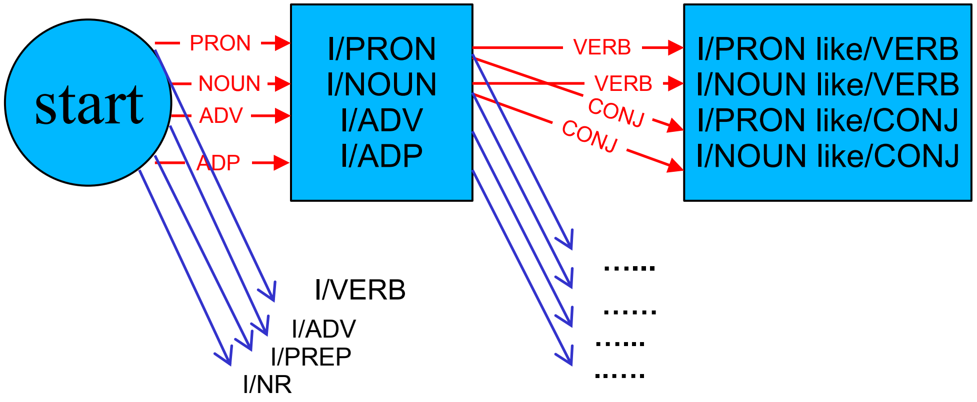
- An example: POS-tagging
  - I like reading books



Zhang and Clark, CL 2011

Beam-search decoding

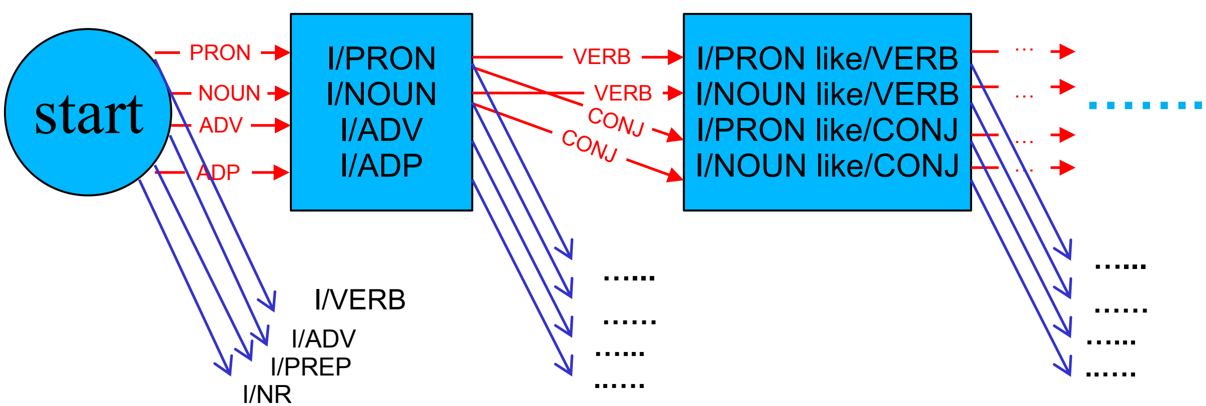
- An example: POS-tagging
  - I like reading books



Zhang and Clark, CL 2011

Beam-search decoding

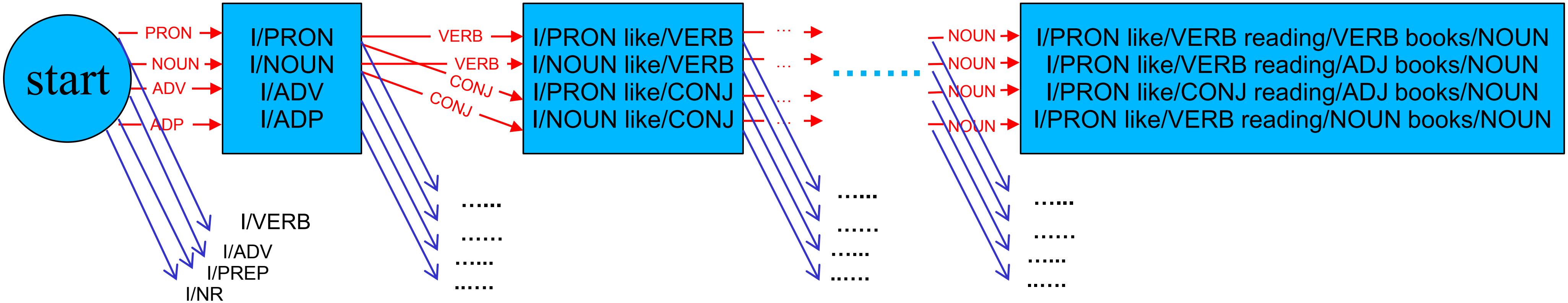
- An example: POS-tagging
  - I like reading books



Zhang and Clark, CL 2011

Beam-search decoding

- An example: POS-tagging
  - I like reading books

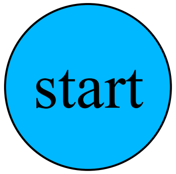


Zhang and Clark, CL 2011

Introduction

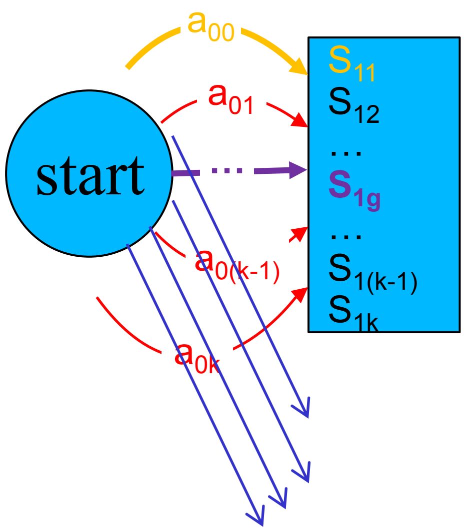
- **Structured prediction problems**
- **An Overview of the transition system**
- **Algorithms in details**
  - **Beam-search decoding**
  - **Online learning using averaged perceptron**

Online learning



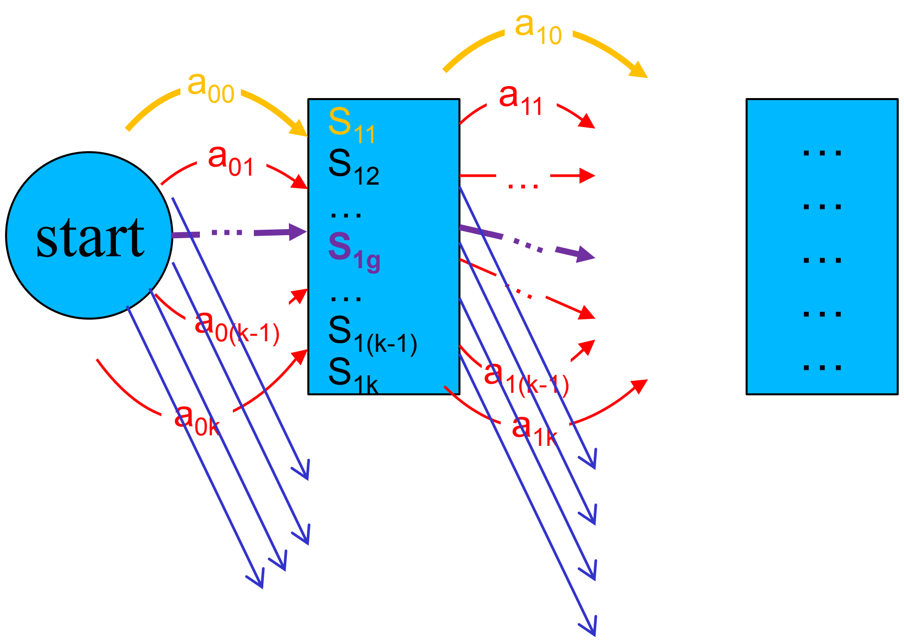
Zhang and Clark, CL 2011

Online learning



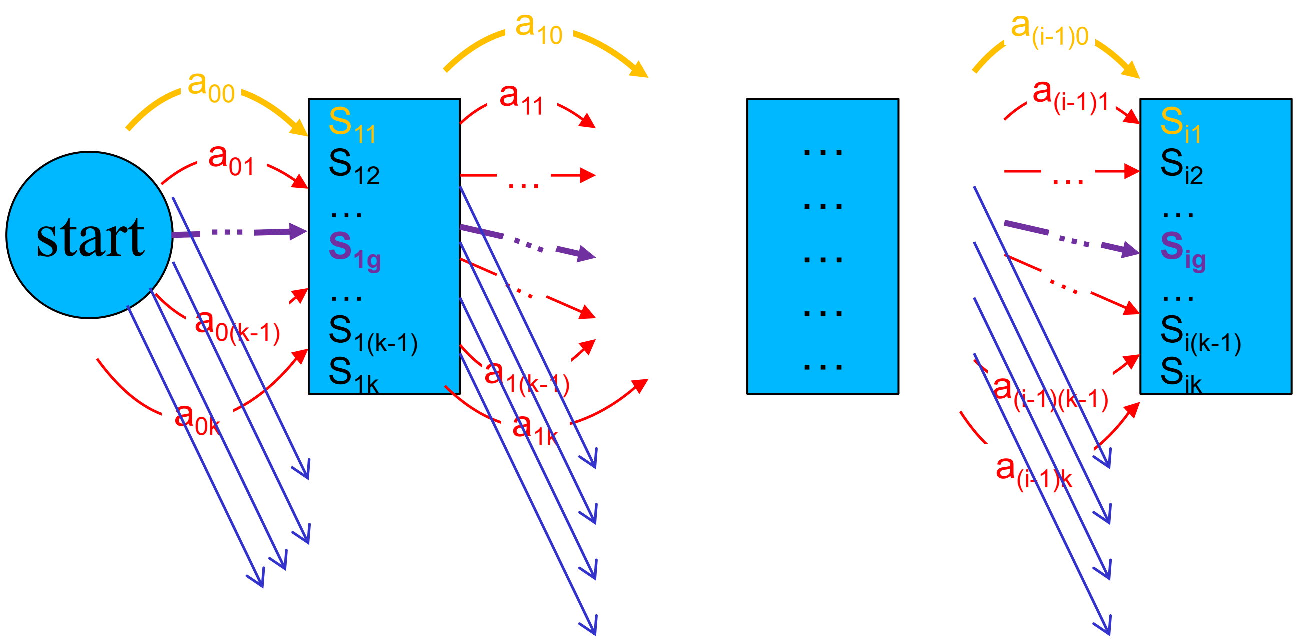
Zhang and Clark, CL 2011

Online learning



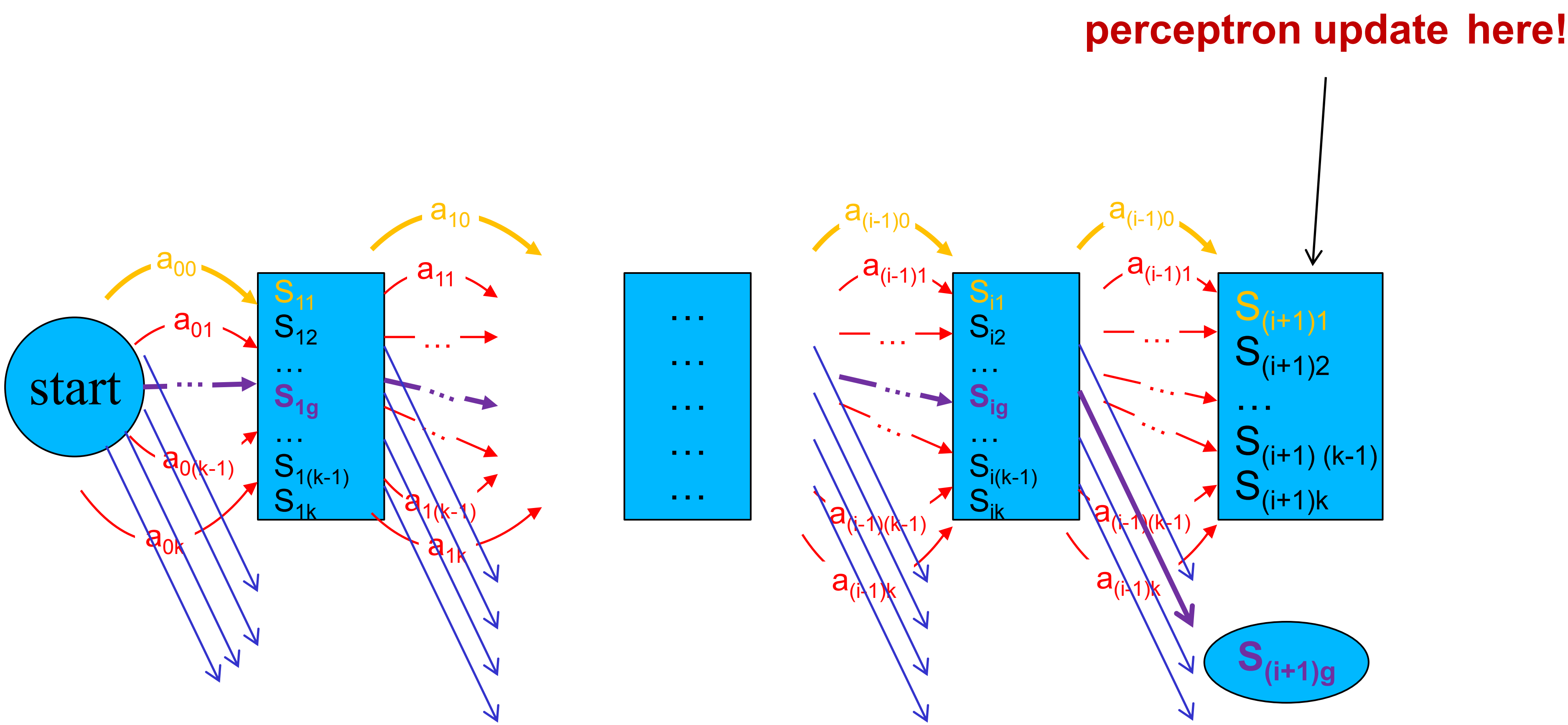
Zhang and Clark, CL 2011

Online learning



Zhang and Clark, CL 2011

Online learning



Zhang and Clark, CL 2011

Online learning

Inputs: training examples $(x_i, y_i = \{S_0^i S_1^i \cdots S_m^i\}$is a state sequence$)_1^N$
Initialization: set $\overrightarrow{w} = 0$
Algorithm:
for $r = 1 \cdots P$, $i = 1 \cdots N$ **do**
  $candidates \leftharpoonup \{S_0^i\}$
  $agenda \leftharpoonup \mathrm{CLEAR}(agenda)$
  for $k = 1 \cdots m$, m corresponds to a specific training example. **do**
    for each $candidate$ in $candidates$ **do**
      $agenda \leftharpoonup \mathrm{INSERT}(\mathrm{EXPAND}(candidate), agenda)$
      $candidates \leftharpoonup \mathrm{TOP} - \mathrm{B}(agenda, B)$
      $best \leftharpoonup \mathrm{TOP}(agenda)$
      if S_k^i is not in $candidates$ or ($best \neq S_m^i$ and k equals m) **then**
        $\overrightarrow{w} = \overrightarrow{w} + \Phi(S_k^i) - \Phi(best)$
      end if
    end for
  end for
end for
Output: $\overrightarrow{w}$

Zhang and Clark, CL 2011

Outline

Introduction	**Applications**
Analysis	ZPar

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

Introduction

- Chinese word segmentation
  我喜欢读书　　　Ilikereadingbooks
  我 喜欢 读 书　　　I like reading books
- Ambiguity
  - Out-of-vocabulary words (OOV words)
    进步 (make progress; OOV)
    进(advance; known) 步(step; known)
  - Known words
    这里面：　这里(here) 面(flour) 很(very) 贵(expensive)
    　　这(here) 里面(inside) 很 (very) 冷 (cold)
  - 洽谈会很成功：

    洽谈会(discussion meeting) 很 (very) 成功(successful)
    洽谈(discussion) 会(will) 很(very) 成功(succeed)

Introduction

- No fixed standard
  - only about 75% agreement among native speakers
  - task dependency

    北京银行: 北京银行(Bank of Beijing)
    北京(Beijing)银行(bank)
- Therefore, supervised learning with specific training corpora seems more appropriate.
- the dominant approach

Introduction

- The character-tagging approach
  - Map word segmentation into character tagging
    我 喜欢 读 书
    我/S喜/B欢/E读/S书/S
  - Context information: neighboring five character window
  - Traditionally CRF is used
  - This method can be implemented using our framework also!

(cf. the sequence labeling example in the intro)

Introduction

- Limitation of the character tagging method
  中国外企业
  其中(among which) 国外(foreign) 企业(companies)
  中国(in China) 外企(foreign companies) 业务 (business)
- Motivation of a word-based method
  - Compare candidates by word information directly
  - Potential for more linguistically motivated features

The transition system

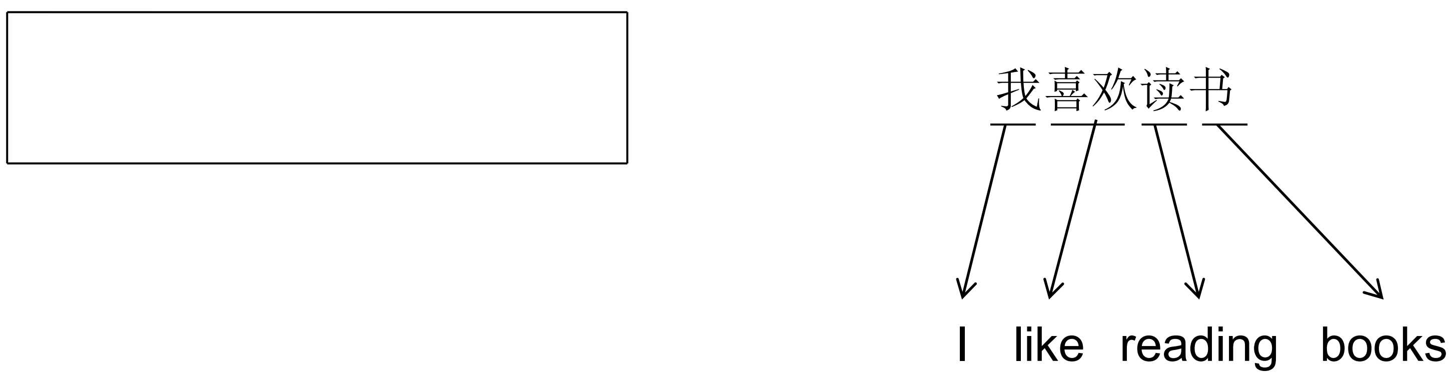
- State
  - Partially segmented results
  - Unprocessed characters

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

Zhang and Clark, ACL 2007

The transition system

- Initial State



The transition system

- Separate

我

喜欢读书

The transition system

- Separate

我　喜

欢读书

The transition system

- Append

我　喜欢

读书

The transition system

- Separate

我 喜欢 读

书

The transition system

- Separate

我 喜欢 读 书

Zhang and Clark, ACL 2007

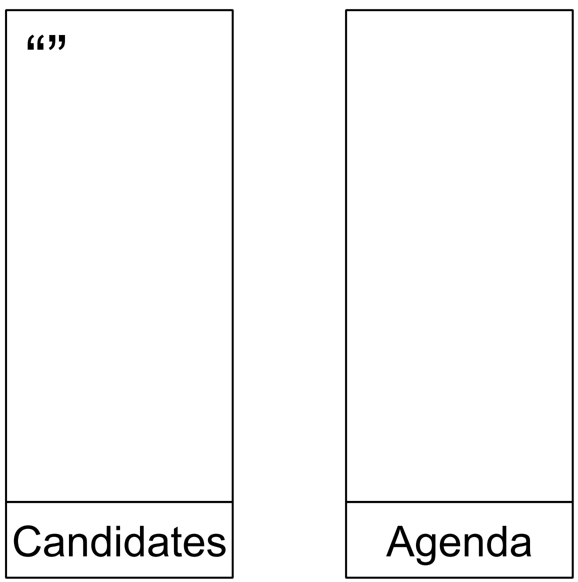
The transition system

- End State

我 喜欢 读 书

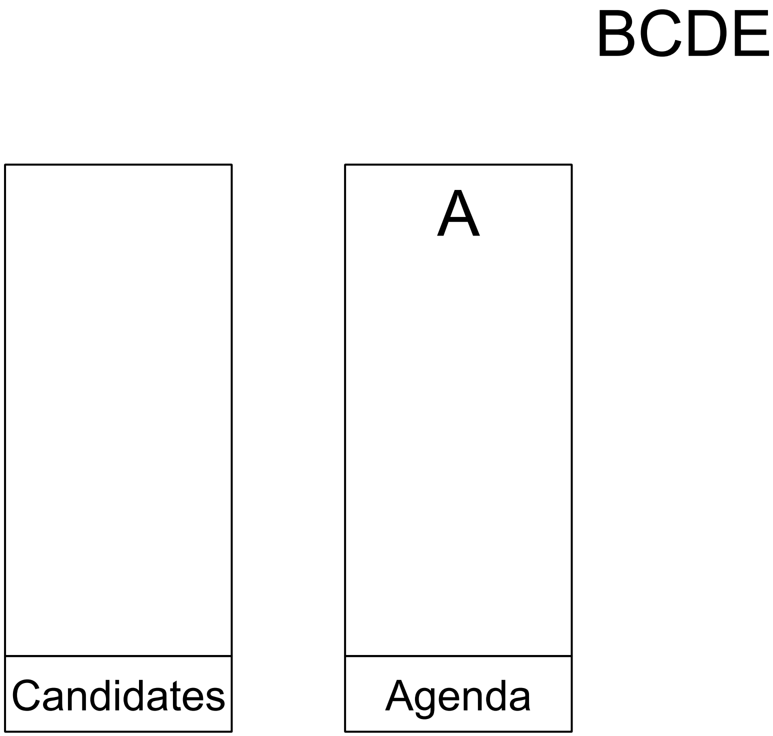
Beam search

ABCDE



Zhang and Clark, ACL 2007

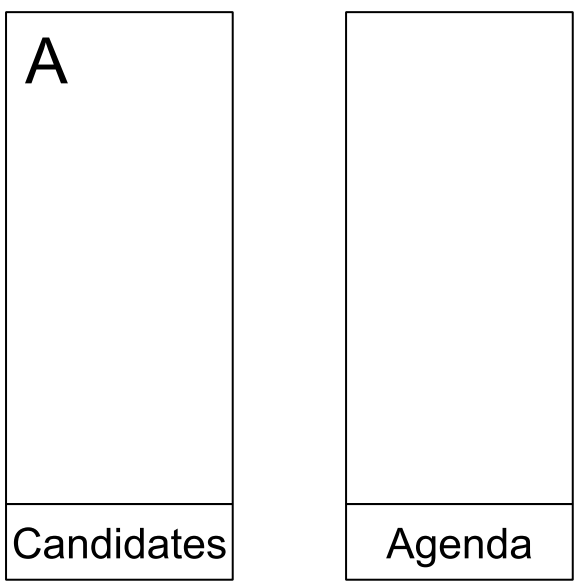
Beam search



Zhang and Clark, ACL 2007

Beam search

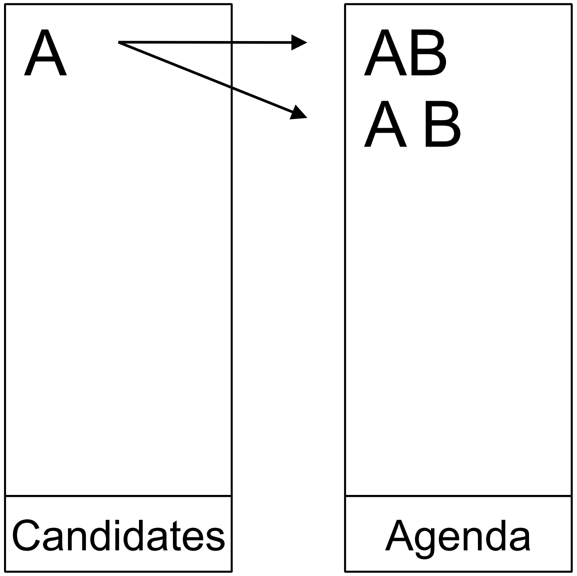
<u>B</u>CDE



Zhang and Clark, ACL 2007

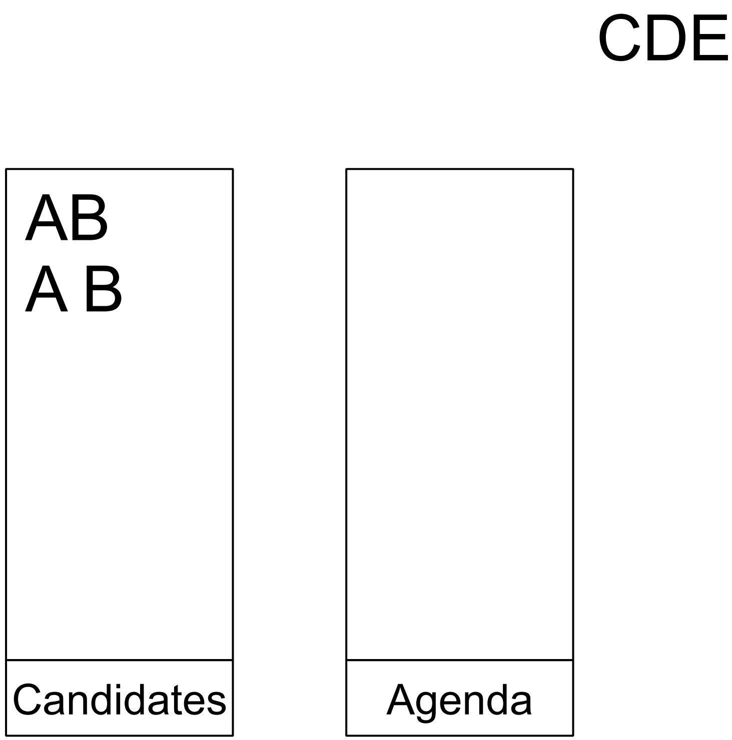
Beam search

CDE



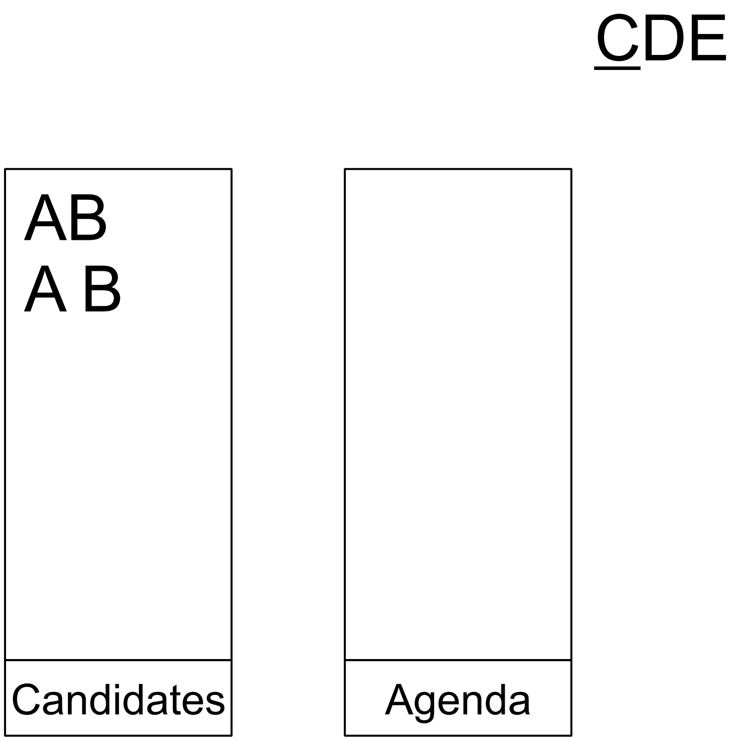
Zhang and Clark, ACL 2007

Beam search



Zhang and Clark, ACL 2007

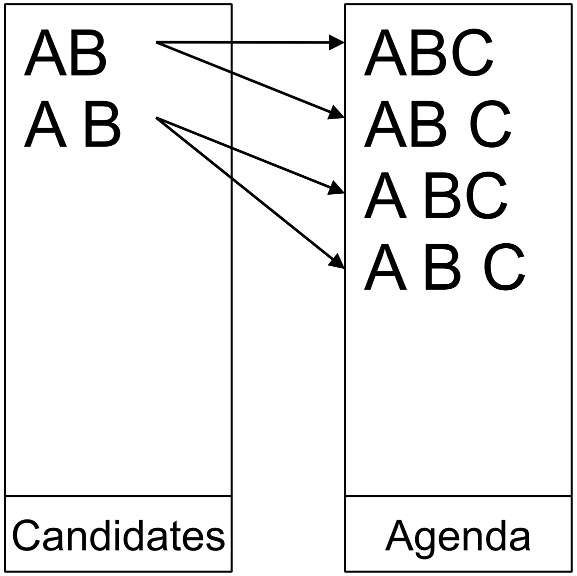
Beam search



Zhang and Clark, ACL 2007

Beam search

DE



Zhang and Clark, ACL 2007

The beam search decoder

- For a given sentence with length=l, there are 2^{l-1} possible segmentations.
- The agenda size is limited, keeping only the *B* best candidates

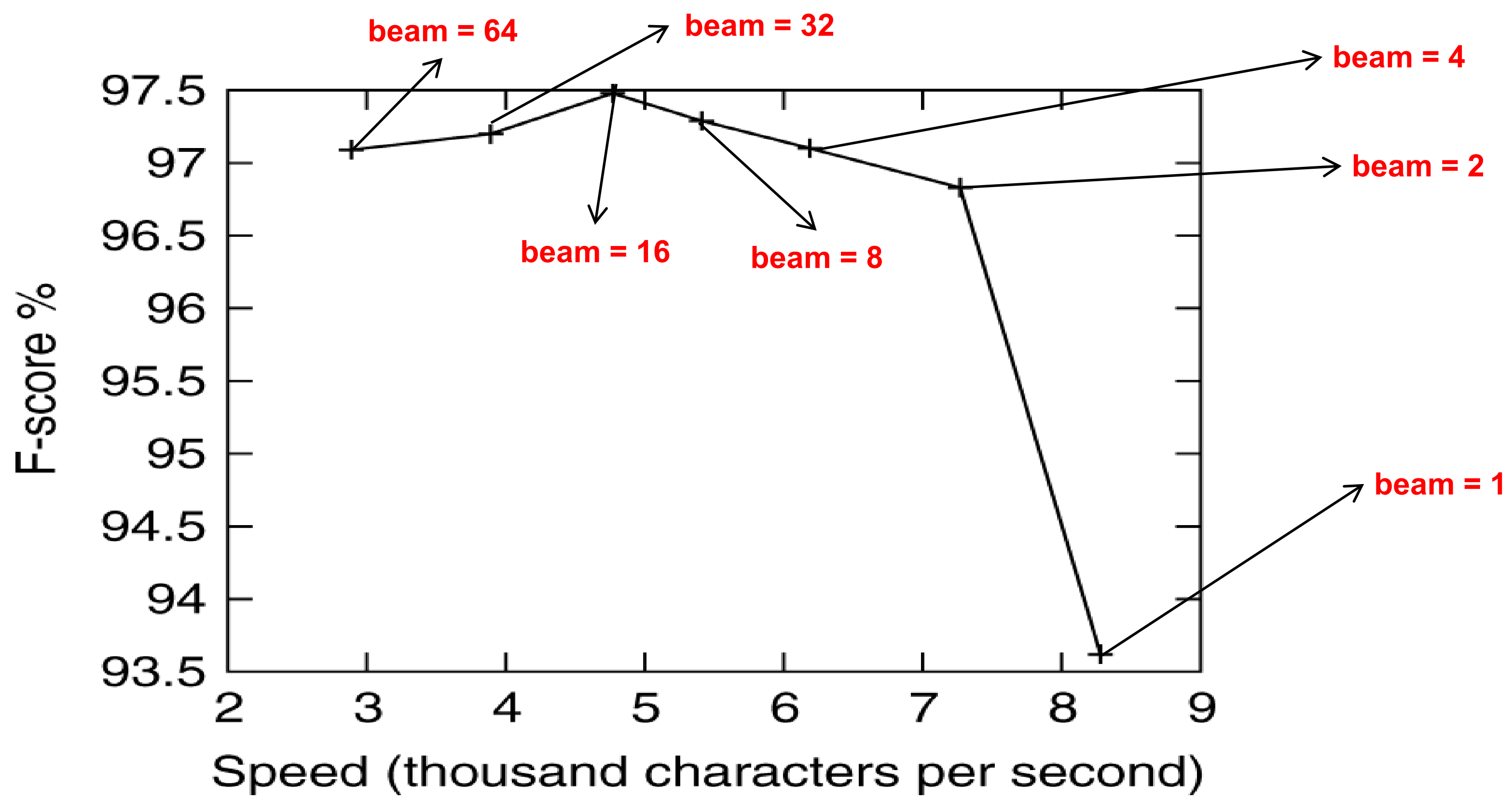
Feature templates

1	word *w*
2	word bigram *w*1*w*2
3	single character word *w*
4	a word starting with character *c* and having length *l*
5	a word ending with character *c* and having length *l*
6	space separated characters *c*1 and *c*2
7	character bigram *c*1*c*2 in any word
8	the first and last characters *c*1 and *c*2 of any word
9	word *w* immediately before character *c*
10	character *c* immediately before word *w*
11	the starting characters *c*1 and *c*2 of two consecutive words
12	the ending characters *c*1 and *c*2 of two consecutive words
13	a word with length *l* and the previous word *w*
14	a word with length *l* and the next word *w*

Zhang and Clark, ACL 2007

Experimental results

- Tradeoff between speed and accuracies (CTB5).



Speed/accuracy tradeoff of the segmentor.

Zhang and Clark, ACL 2007

Experimental results

- Compare with other systems (SIGHAN 2005).

	AS	CU	PU	SAV	OAV
S01	93.8	90.1	95.1	93.0	95.5
S04			93.9	93.9	94.8
S05	94.2		89.4	91.8	95.9
S06	94.5	92.4	92.4	93.1	95.5
S08		90.4	93.6	92.9	94.8
S09	96.1		94.6	95.4	95.9
S10			94.7	94.7	94/8
S12	95.9	91.6		93.8	95.9
Peng	95.6	92.8	94.1	94.2	95.5
Z&C 07	97.0	94.6	94.6	95.4	95.5

Zhang and Clark, ACL 2007

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

Dependency syntax

- Dependency structures represent syntactic relations (**dependencies**) by drawing links between word pairs in a sentence.

- For the link: **a telescope**

  - Modifier
  - Dependent
  - Child

  - Head
  - Governor
  - Parent

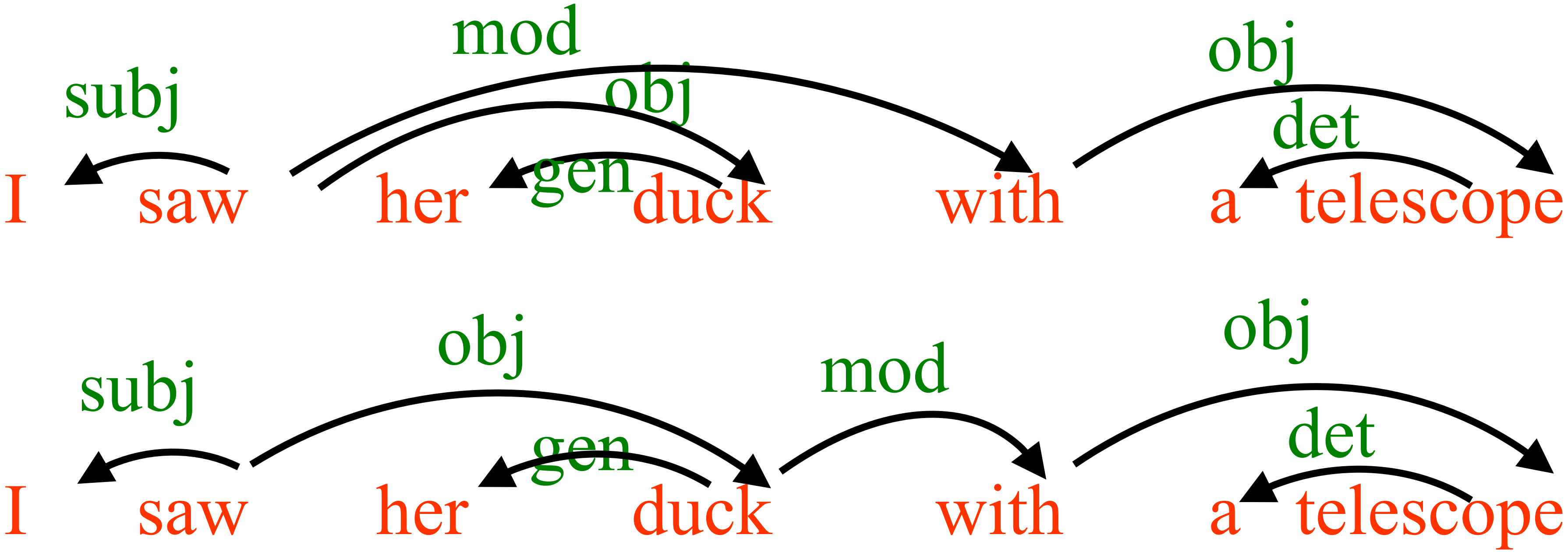
Dependency graphs

- A dependency structure is a directed graph G with the following constraints:
  - Connected
  - Acyclic
  - Single-head

tree

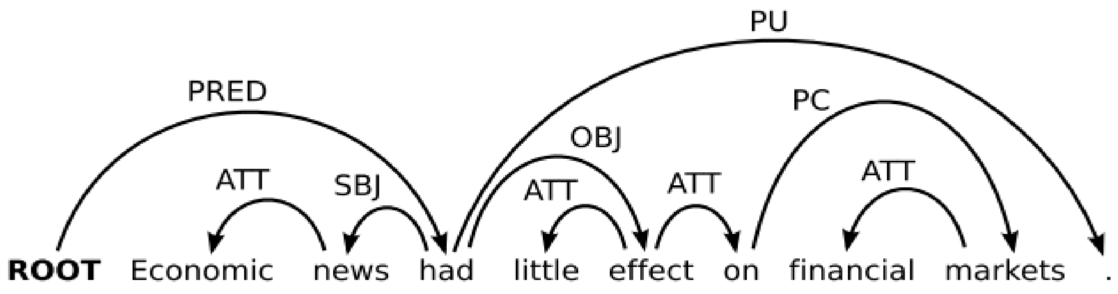
Dependency trees

- A dependency tree structure represents syntactic relations between word pairs in a sentence

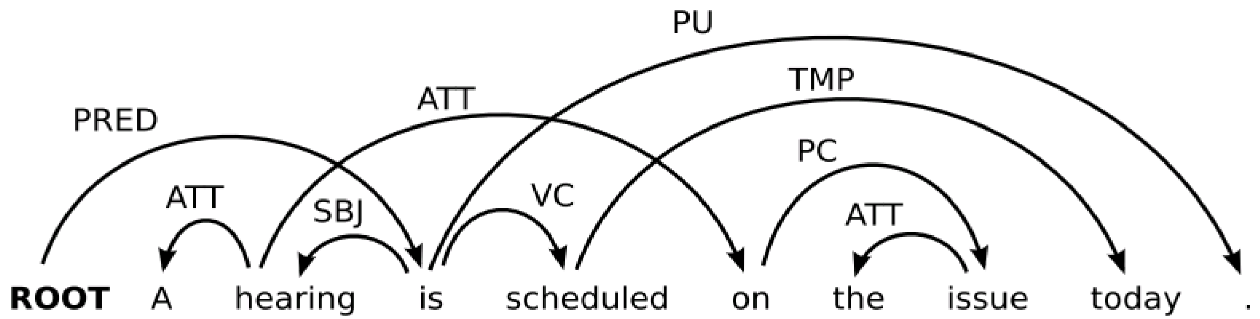


Dependency trees

- Categorization (Kübler et al. 2009)
  - Projective

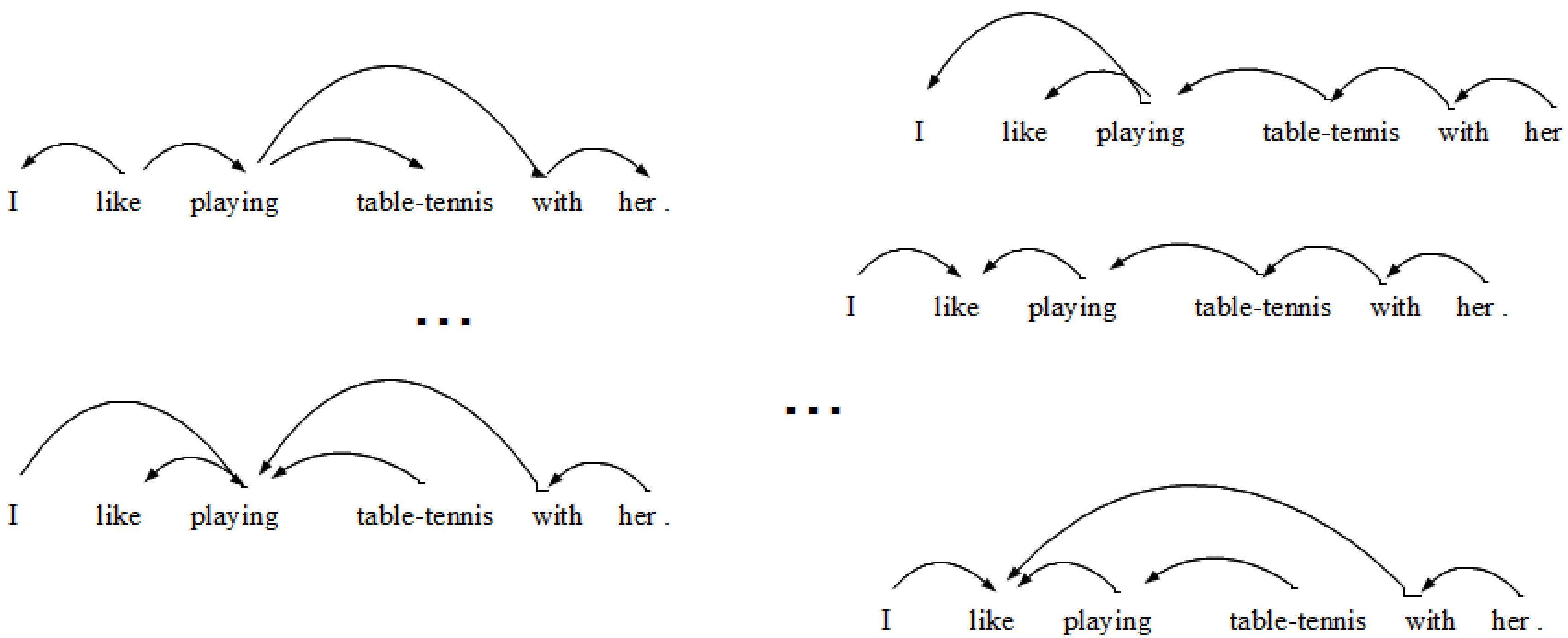


  - Non-projective



The graph-based solution

- Score each possible output
- Often use dynamic programming to explore search space



McDonald et al., ACL 2005
Carreras, EMNLP-CONLL 2007;
Koo and Collins, ACL 2010

Transition systems

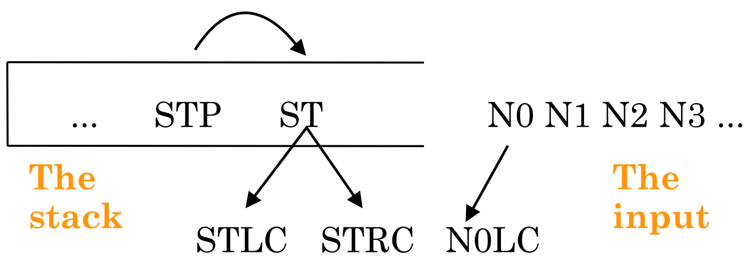
- Projective
  - Arc-eager
  - Arc-standard (Nirve, CL 2008)
- Non-projective
  - Arc standard + swap (Nirve, ACL 2009)

The arc-eager transition system

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

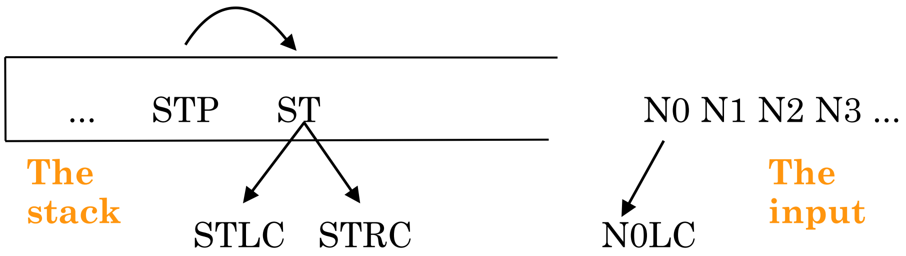
The arc-eager transition system

- State



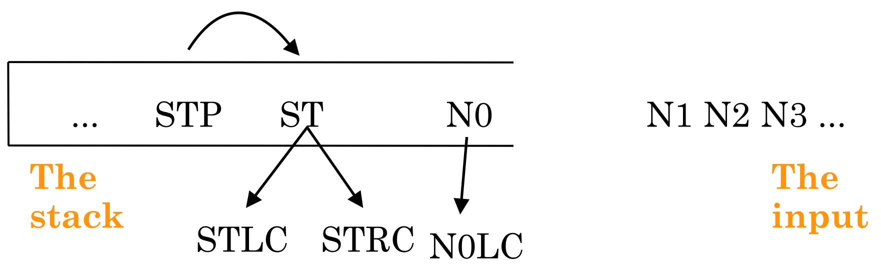
The arc-eager transition system

- Actions
  - Shift



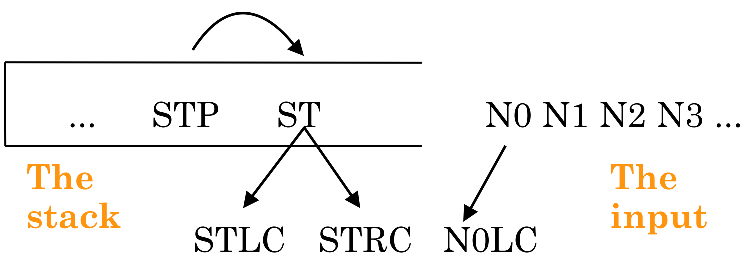
The arc-eager transition system

- Actions
  - Shift
    - Pushes stack



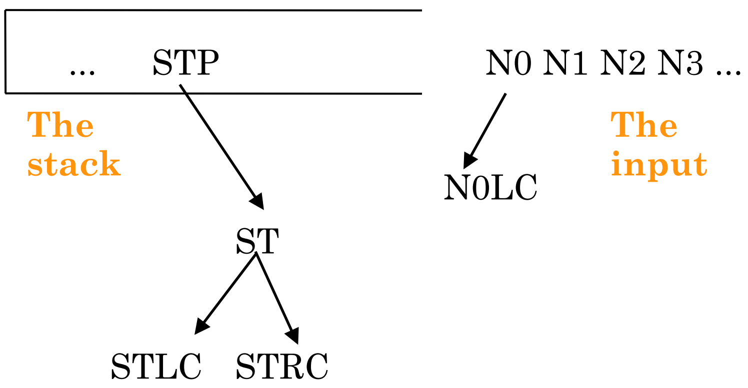
The arc-eager transition system

- Actions
  - Reduce



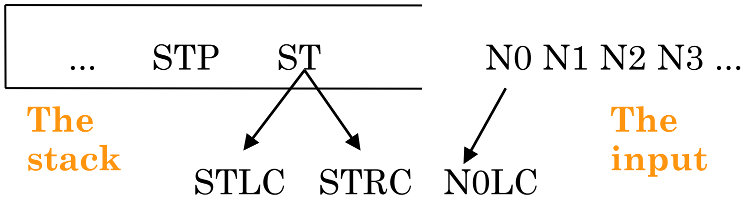
The arc-eager transition system

- Actions
  - Reduce
    - Pops stack



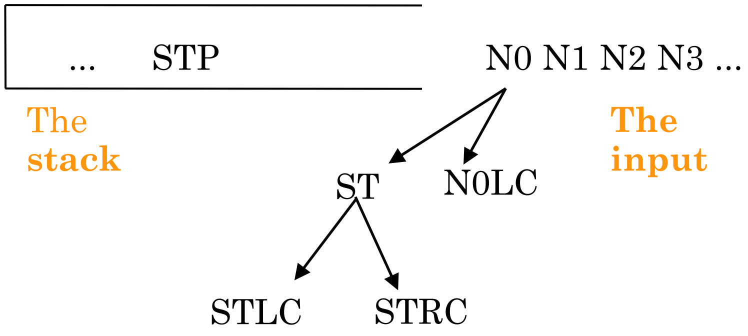
The arc-eager transition system

- Actions
  - Arc-Left



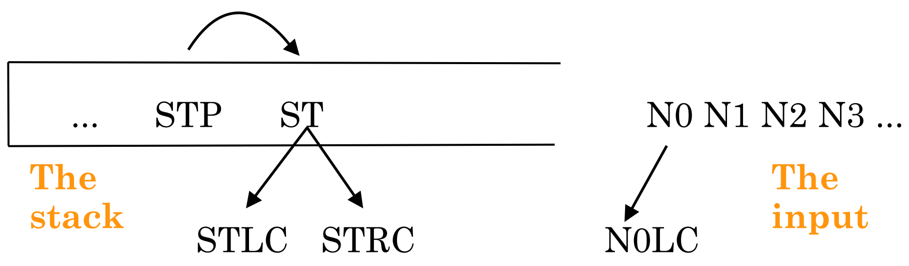
The arc-eager transition system

- Actions
  - Arc-Left
    - Pops stack
    - Adds link



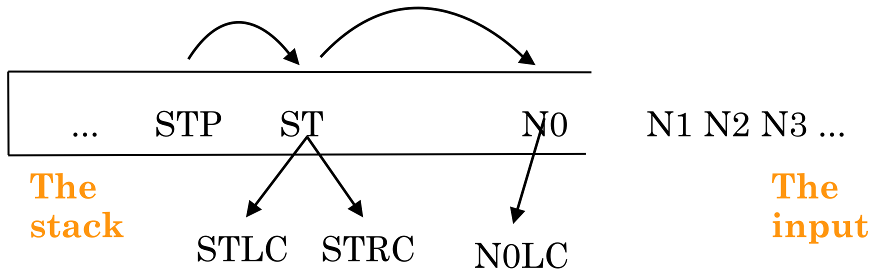
The arc-eager transition system

- Actions
  - Arc-right



The arc-eager transition system

- Actions
  - Arc-right
    - Pushes stack
    - Adds link



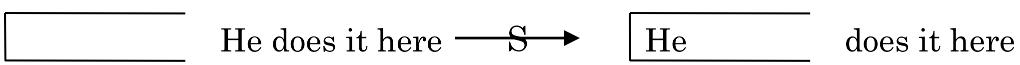
The arc-eager transition system

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

He does it here

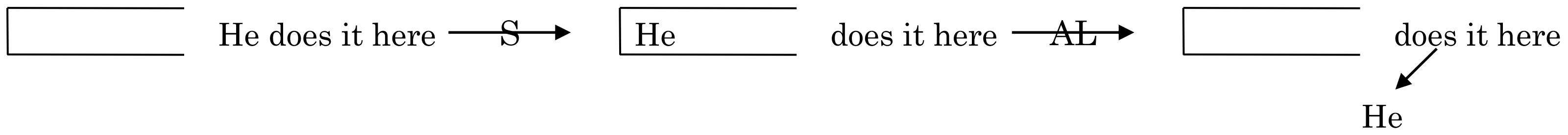
The arc-eager transition system

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



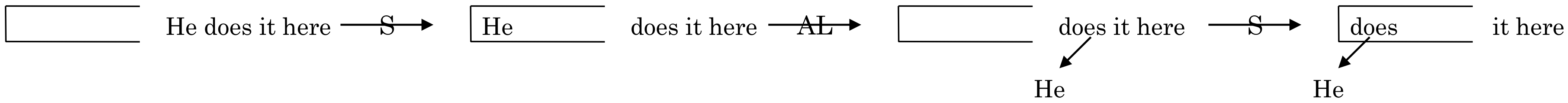
The arc-eager transition system

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



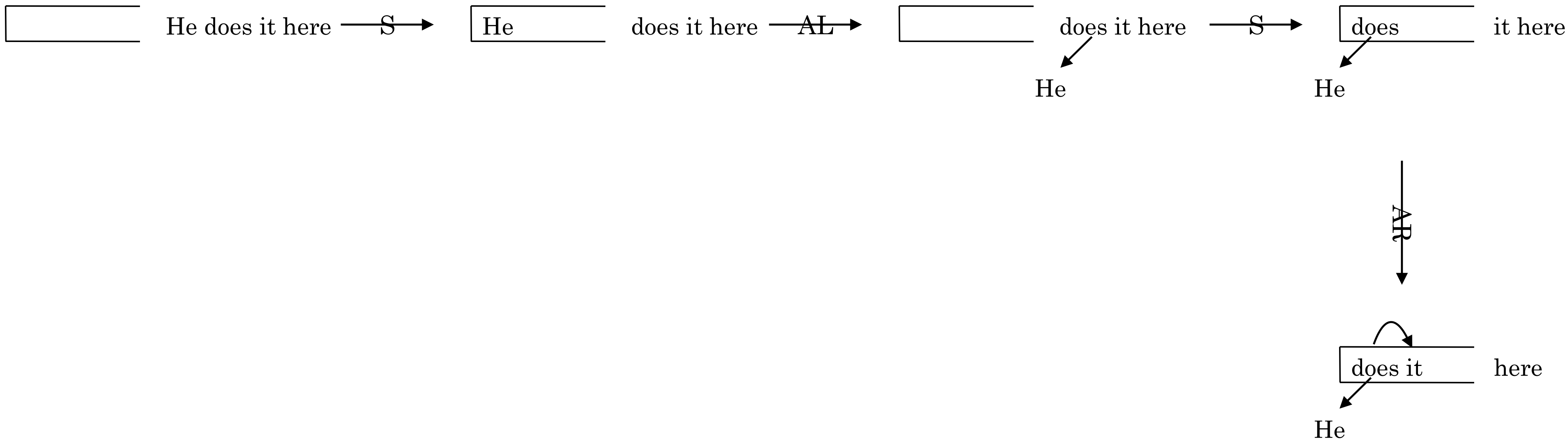
The arc-eager transition system

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



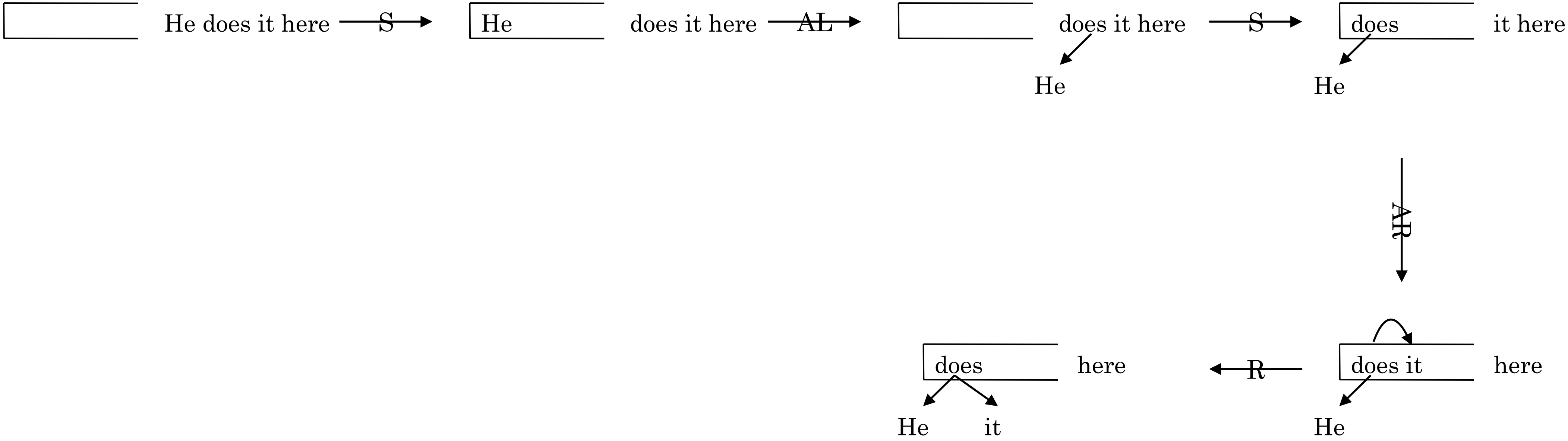
The arc-eager transition system

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



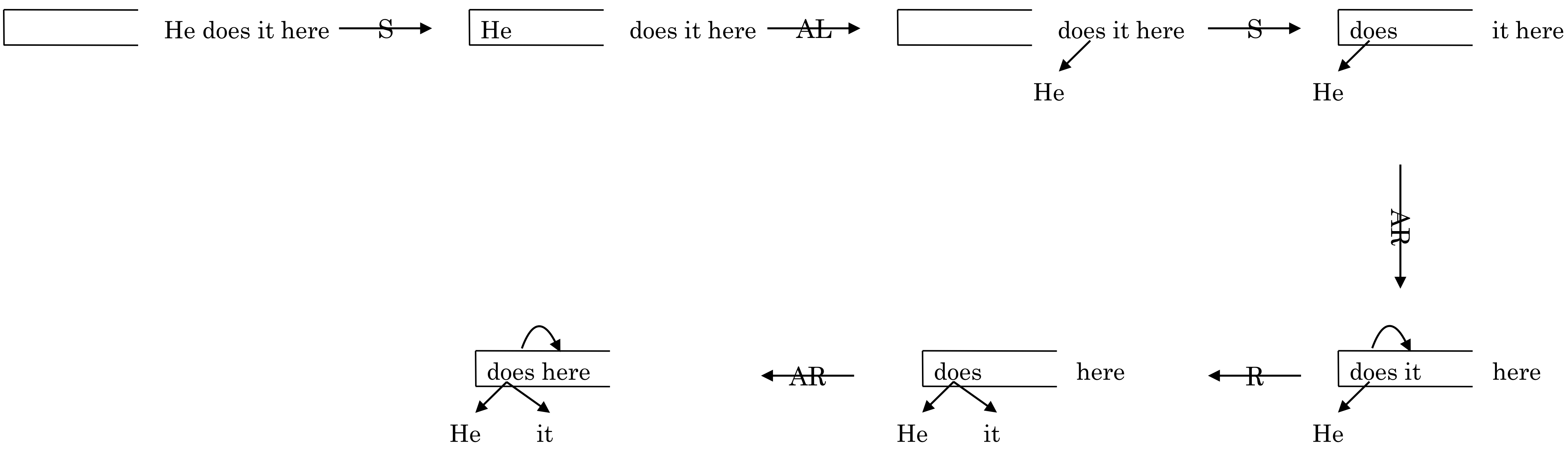
The arc-eager transition system

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



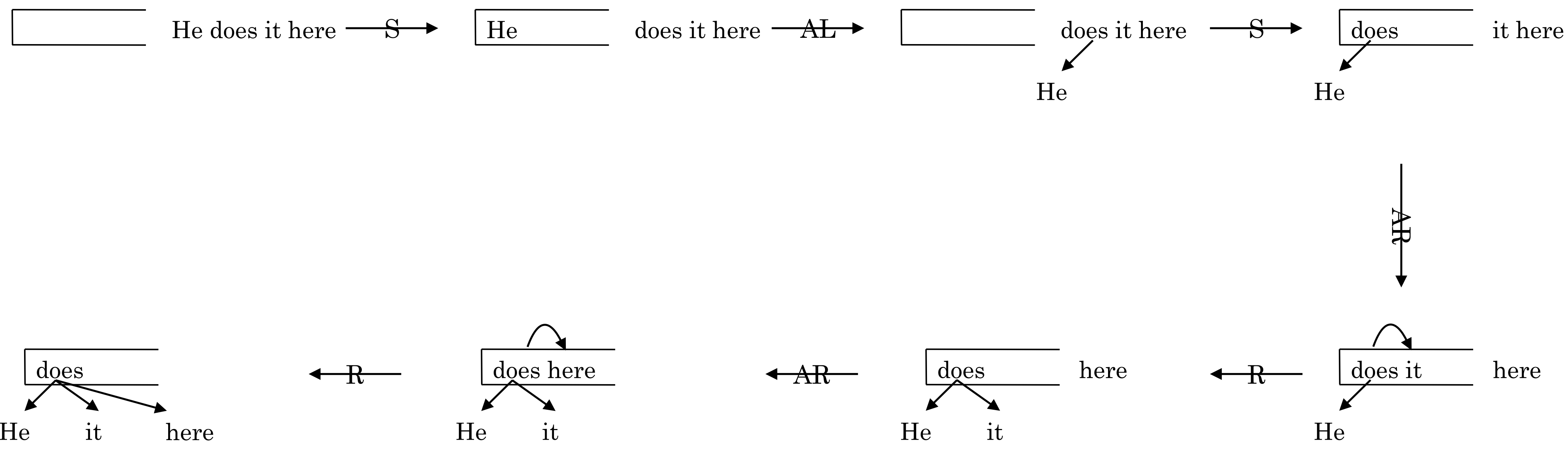
The arc-eager transition system

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



The arc-eager transition system

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



The arc-eager transition system

- Arc-eager
  - Time complexity: linear
    - Every word is pushed once onto the stack
    - Every word except the root is popped once
  - Links are added between ST and N0
    - As soon as they are in place
    - 'eager'

The arc-eager transition system

- Arc-eager
  - Labeled parsing? – expand the link-adding actions

ArcLeft

ArcLeft subject
ArcLeft noun modifier
…

ArcRight

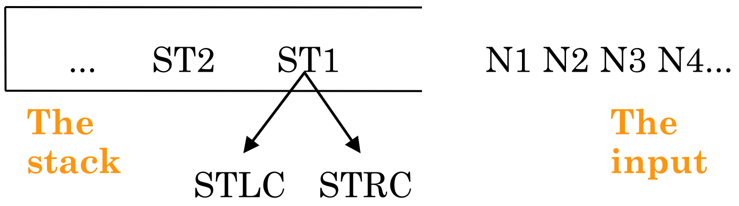
ArcRight object
ArcRight prep modifier
…

The arc-standard transition system

- State
  - A stack to hold partial candidates
  - A queue of next incoming words
- Actions
  - SHIFT LEFT-REDUCE RIGHT-REDUCE
  - Builds arcs between ST0 and ST1
  - Associated with shift-reduce CFG parsing process

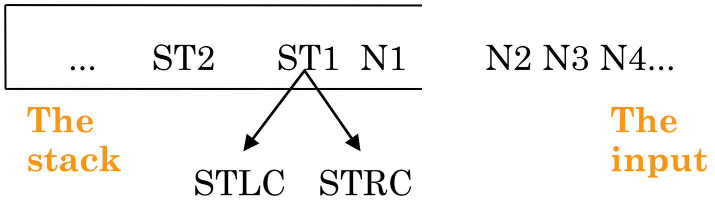
The arc-standard transition system

- Actions
  - Shift



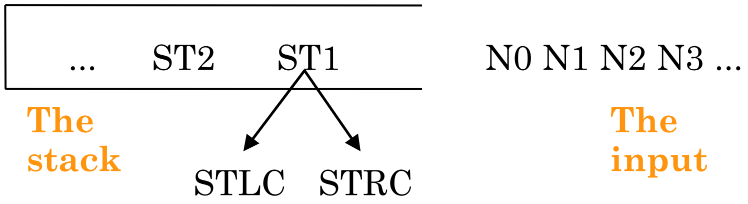
The arc-standard transition system

- Actions
  - Shift
    - Pushes stack



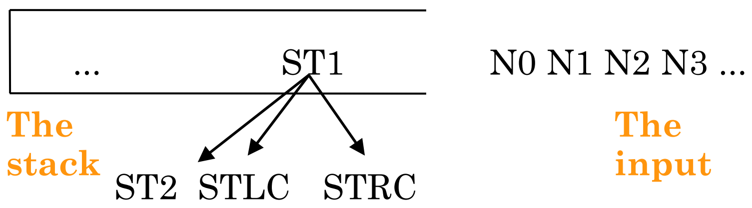
The arc-standard transition system

- Actions
  - Left-reduce



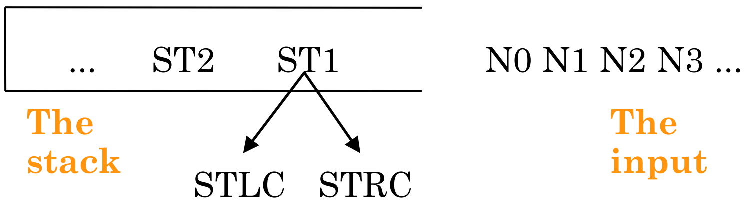
The arc-standard transition system

- Actions
  - Left-reduce
    - Pops stack
    - Adds link



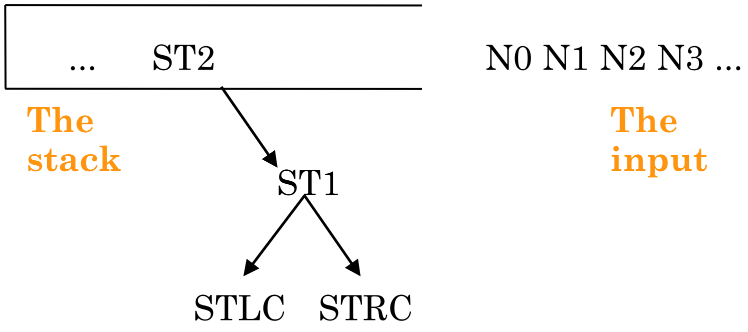
The arc-standard transition system

- Actions
  - Right-reduce



The arc-standard transition system

- Actions
  - Right-reduce
    - Pops stack
    - Adds link

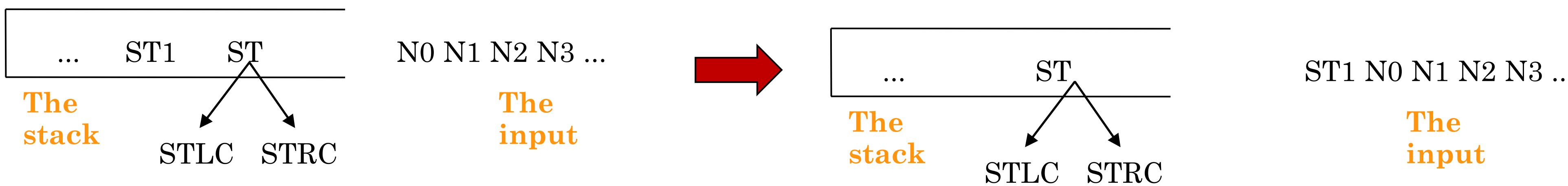


The arc-standard transition system

- Characteristic
  - Time complexity: linear
  - Empirically comparable with arc-eager, but accuracies for different languages are different

Non-projectivity

- Online reordering (Nivre 2009)
  - Based on an extra action to the parser: swap



  - Not linear any more
    - Can be quadratic due to swap
    - Expected linear time

Non-projectivity

- Initial

A meeting was scheduled for this today

Non-projectivity

- SHIFT

A

meeting was scheduled for this today

Non-projectivity

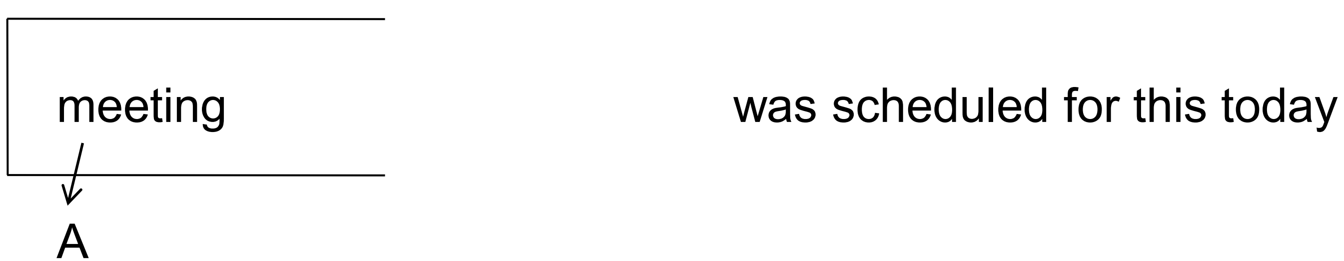
- SHIFT

A meeting

was scheduled for this today

A transition-based parsing process

- ARC-LEFT

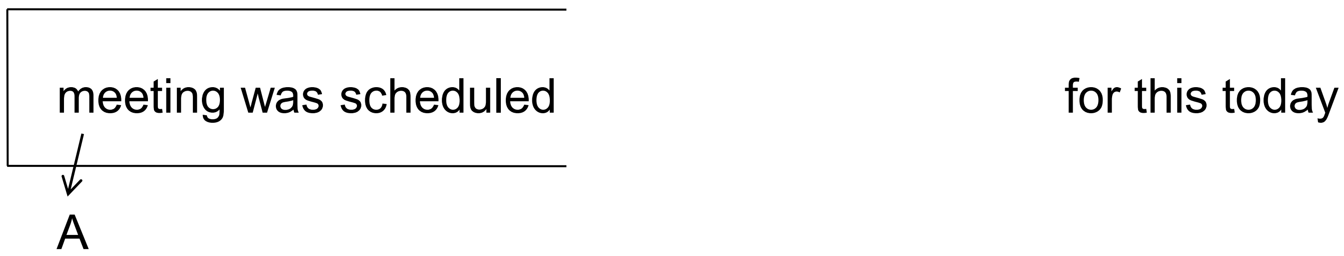


A transition-based parsing process

- SHIFT

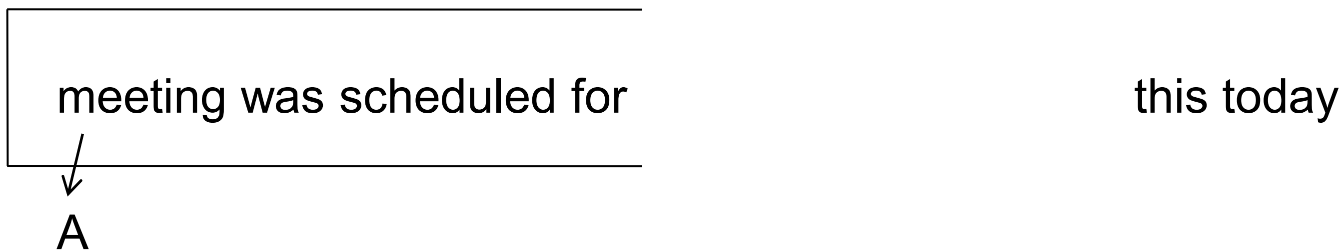
A transition-based parsing process

- SHIFT



A transition-based parsing process

- SHIFT



A transition-based parsing process

- **SWAP**

meeting was **for** scheduled this today

A

A transition-based parsing process

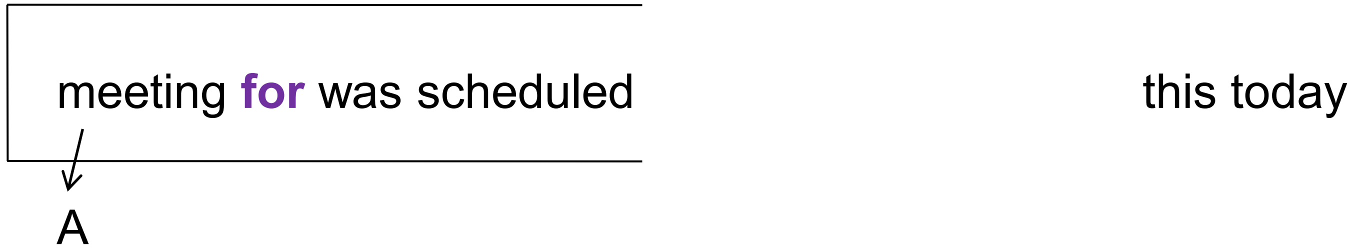
- **SWAP**

A transition-based parsing process

- **SHIFT**

A transition-based parsing process

- **SHIFT**



A transition-based parsing process

- **SHIFT**

meeting **for** was scheduled this today

A

A transition-based parsing process

- **SWAP**

meeting **for** was **this**

A

scheduled today

A transition-based parsing process

- **SWAP**

A transition-based parsing process

- **ARC-RIGHT**

A transition-based parsing process

- **ARC-RIGHT**

A transition-based parsing process

- **SHIFT**

A transition-based parsing process

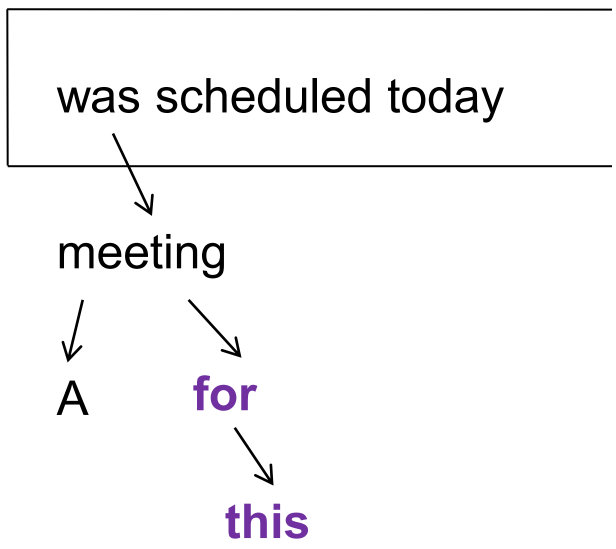
- **ARC-LEFT**

A transition-based parsing process

- **SHIFT**

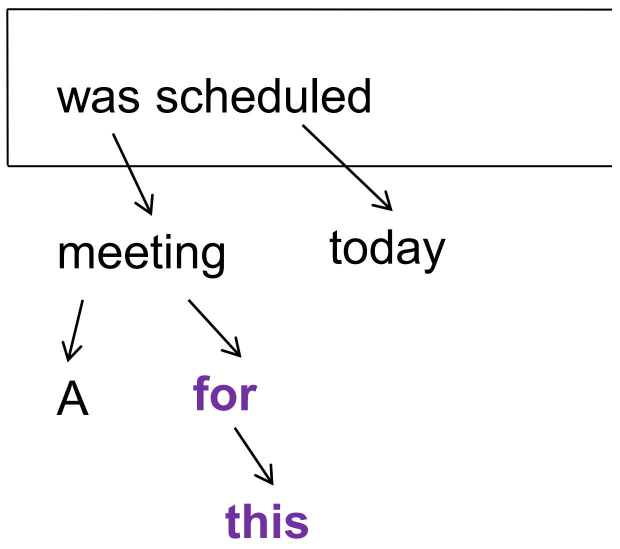
A transition-based parsing process

- **SHIFT**



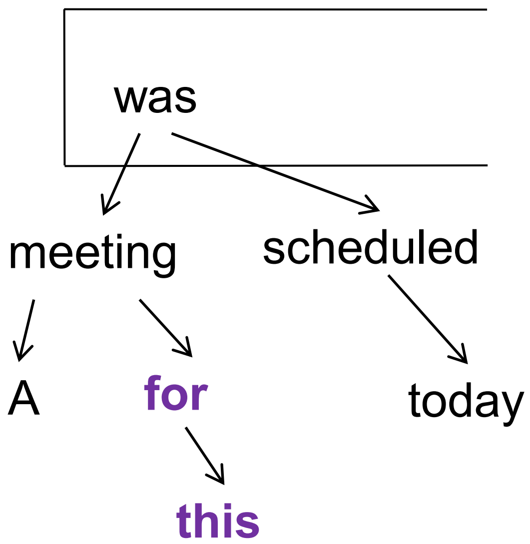
A transition-based parsing process

- **ARC-RIGHT**



A transition-based parsing process

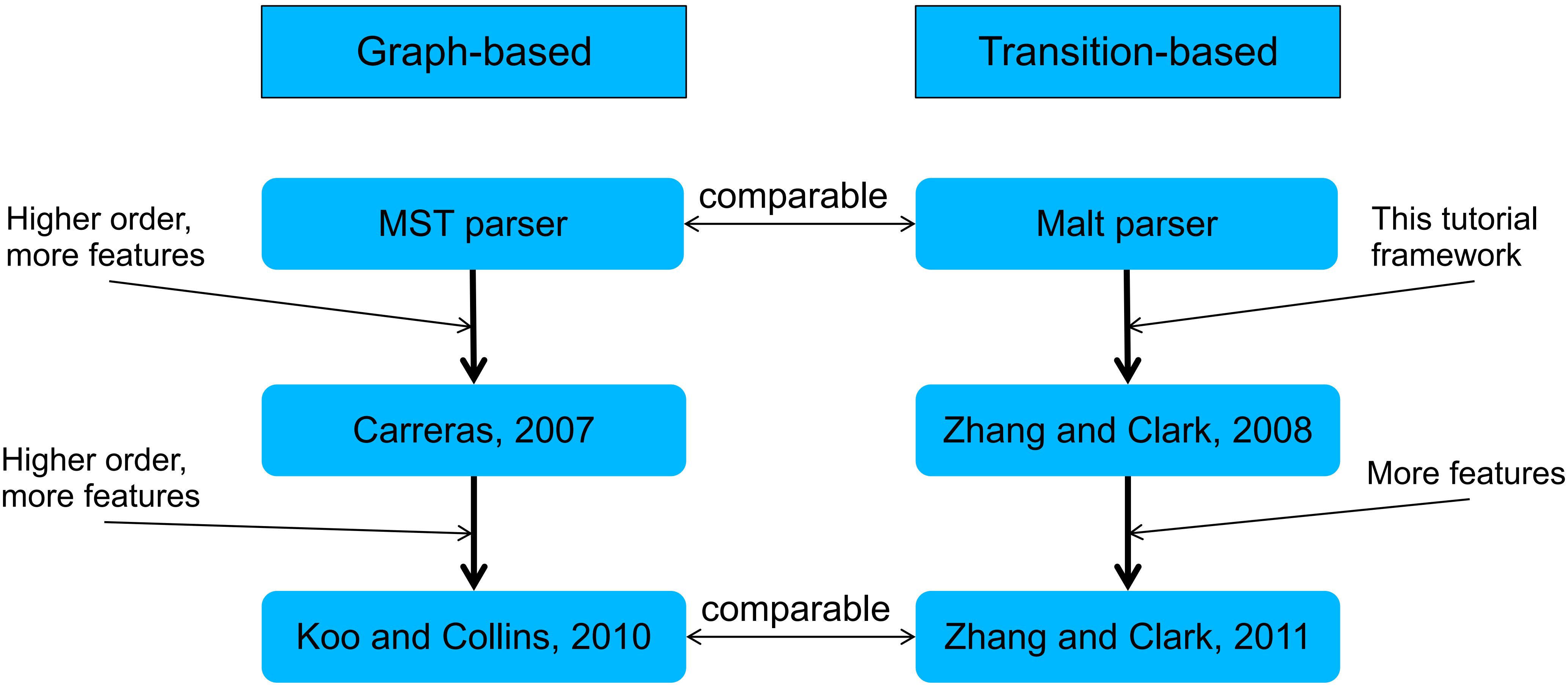
- **ARC-RIGHT**



The arc-eager parser using our framework

- The arc-eager transition process
- Beam-search decoding
  - Keeps N different partial state items in agenda.
  - Use the total score of all actions to rank state items
  - Avoid error propagations from early decisions
- Global discriminative training

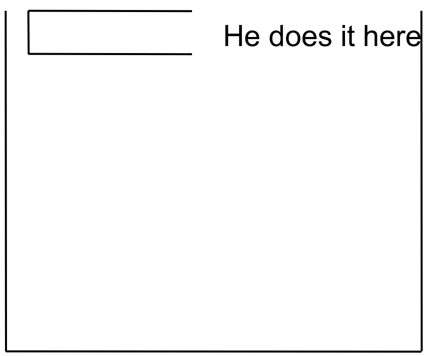
A tale of two parsers



Zhang and Clark, EMNLP 2008

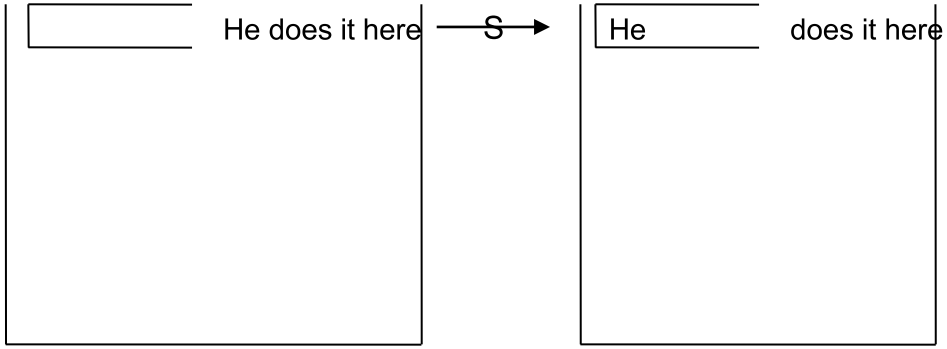
Beam-search decoding

- Our parser
  - Decoding



Beam-search decoding

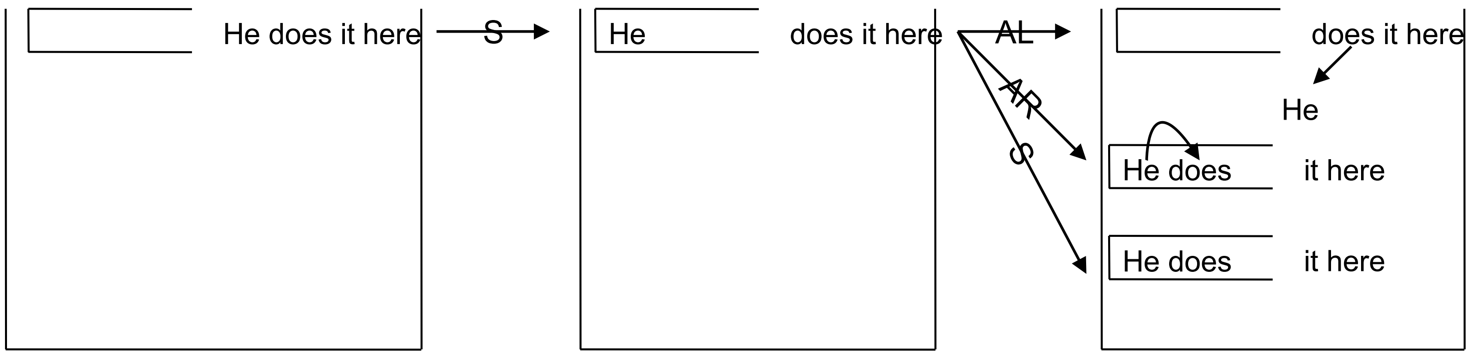
- Our parser
  - Decoding



Zhang and Clark, EMNLP 2008

Beam-search decoding

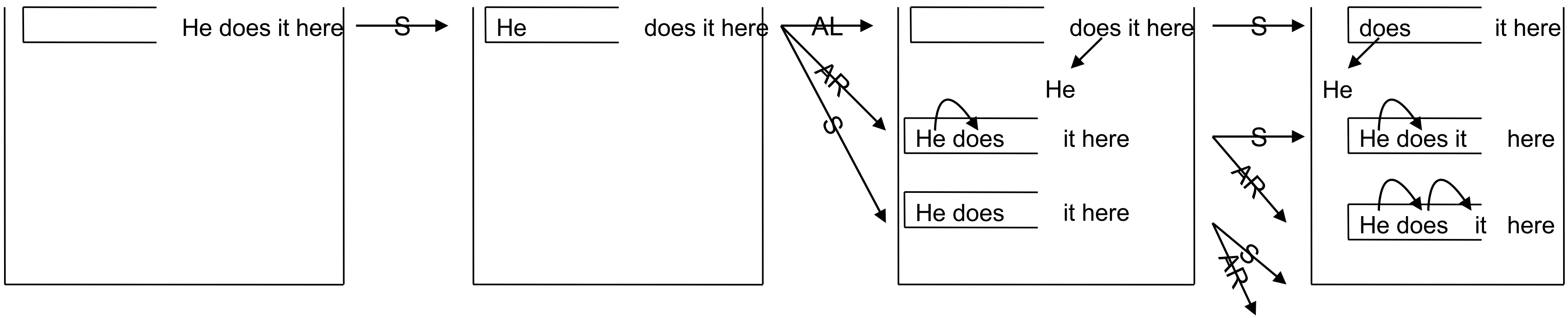
- Our parser
  - Decoding



Zhang and Clark, EMNLP 2008

Beam-search decoding

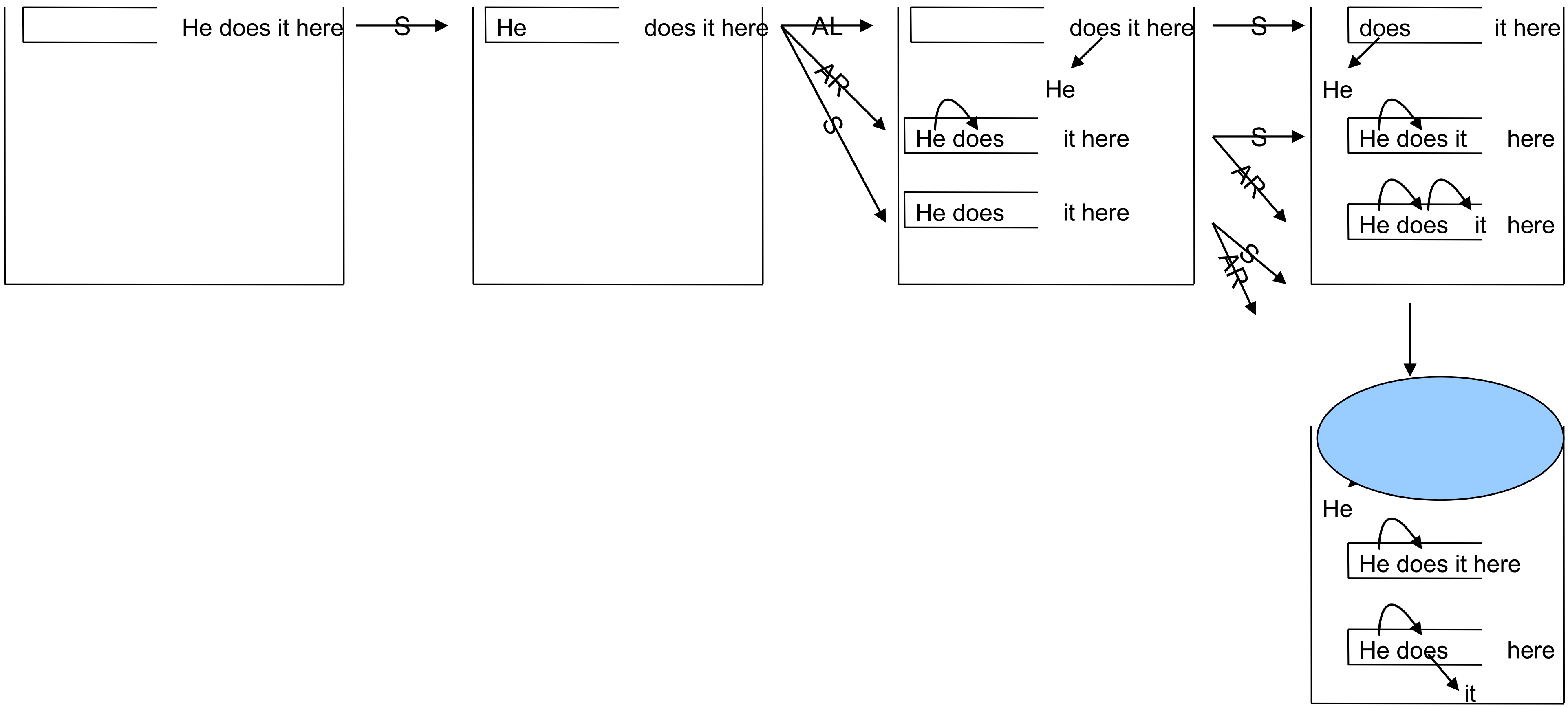
- Our parser
  - Decoding



Zhang and Clark, EMNLP 2008

Beam-search decoding

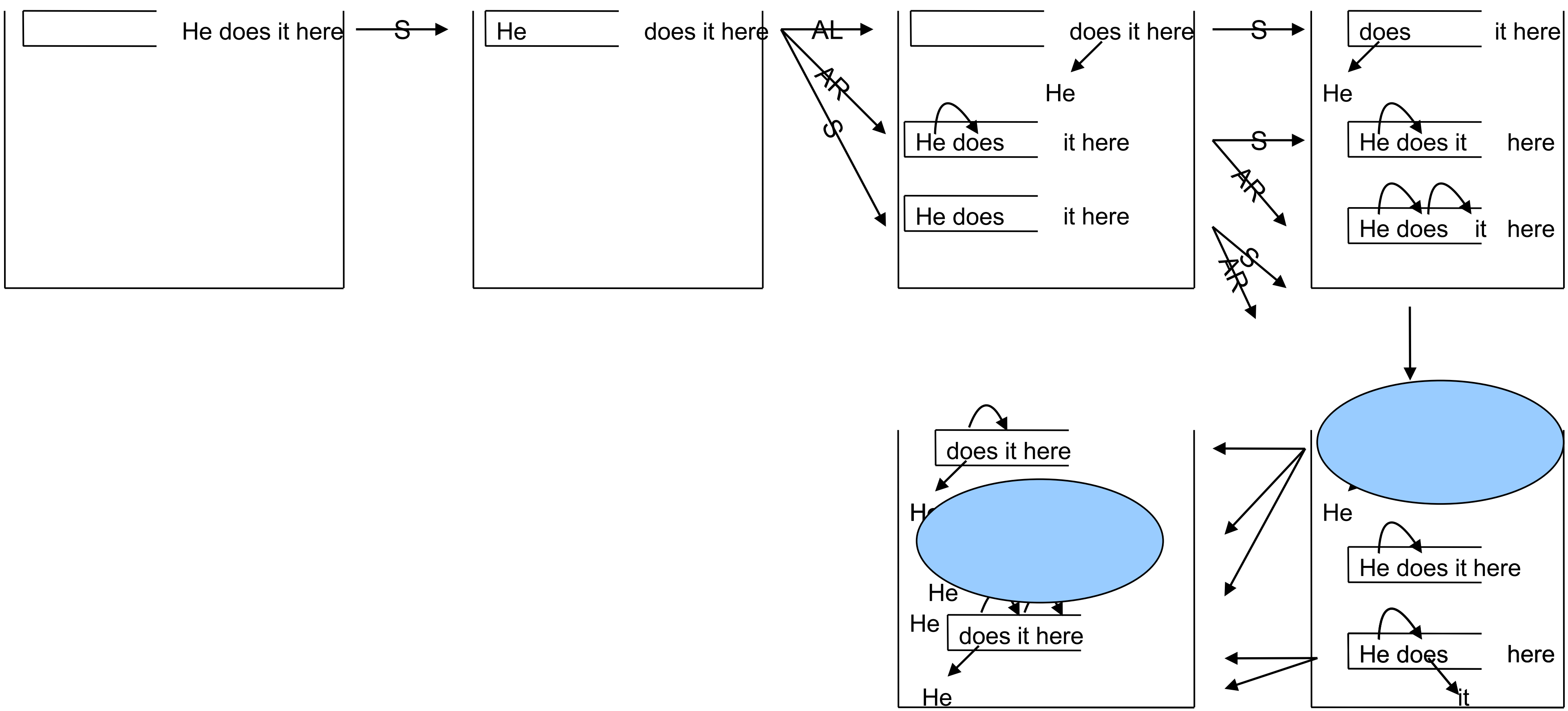
- Our parser
  - Decoding



Zhang and Clark, EMNLP 2008

Beam-search decoding

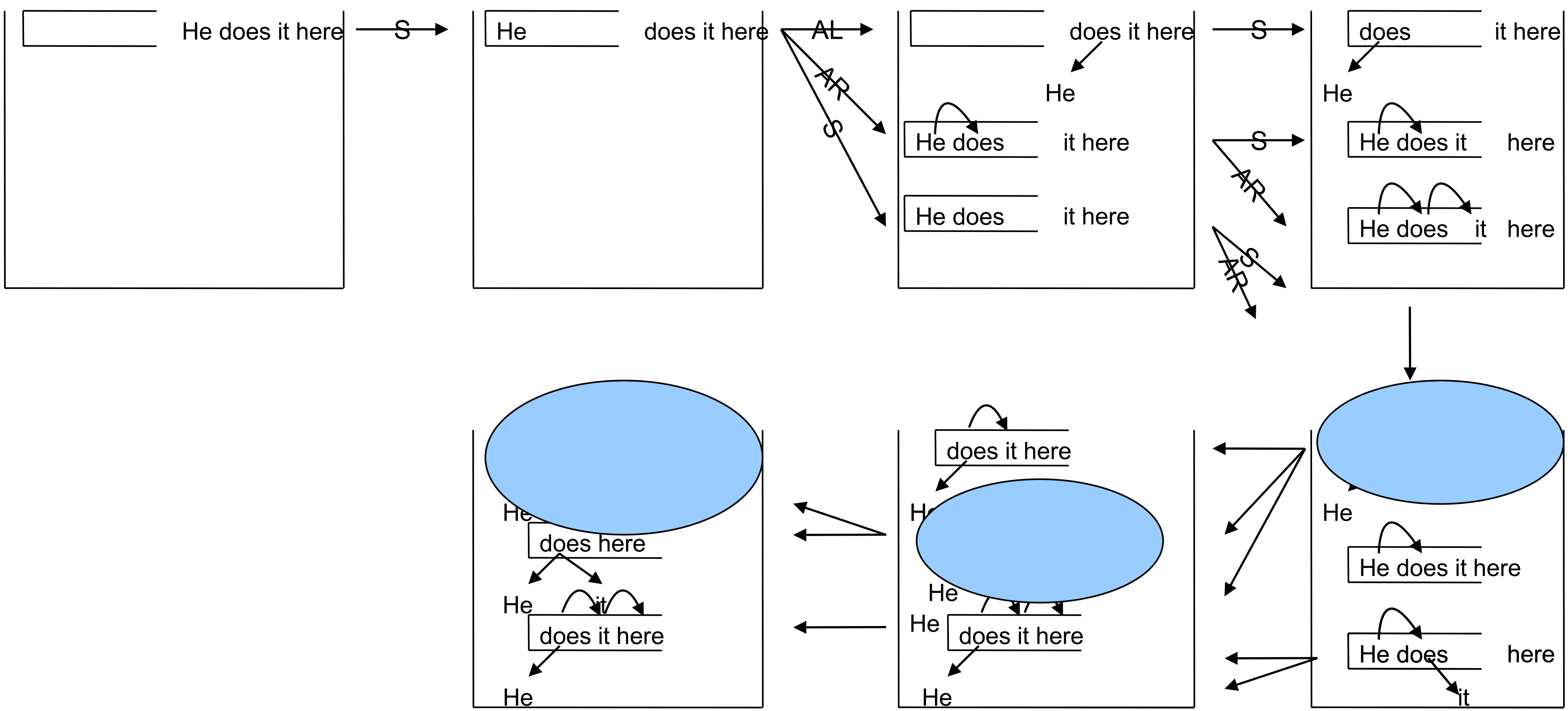
- Our parser
  - Decoding



Zhang and Clark, EMNLP 2008

Beam-search decoding

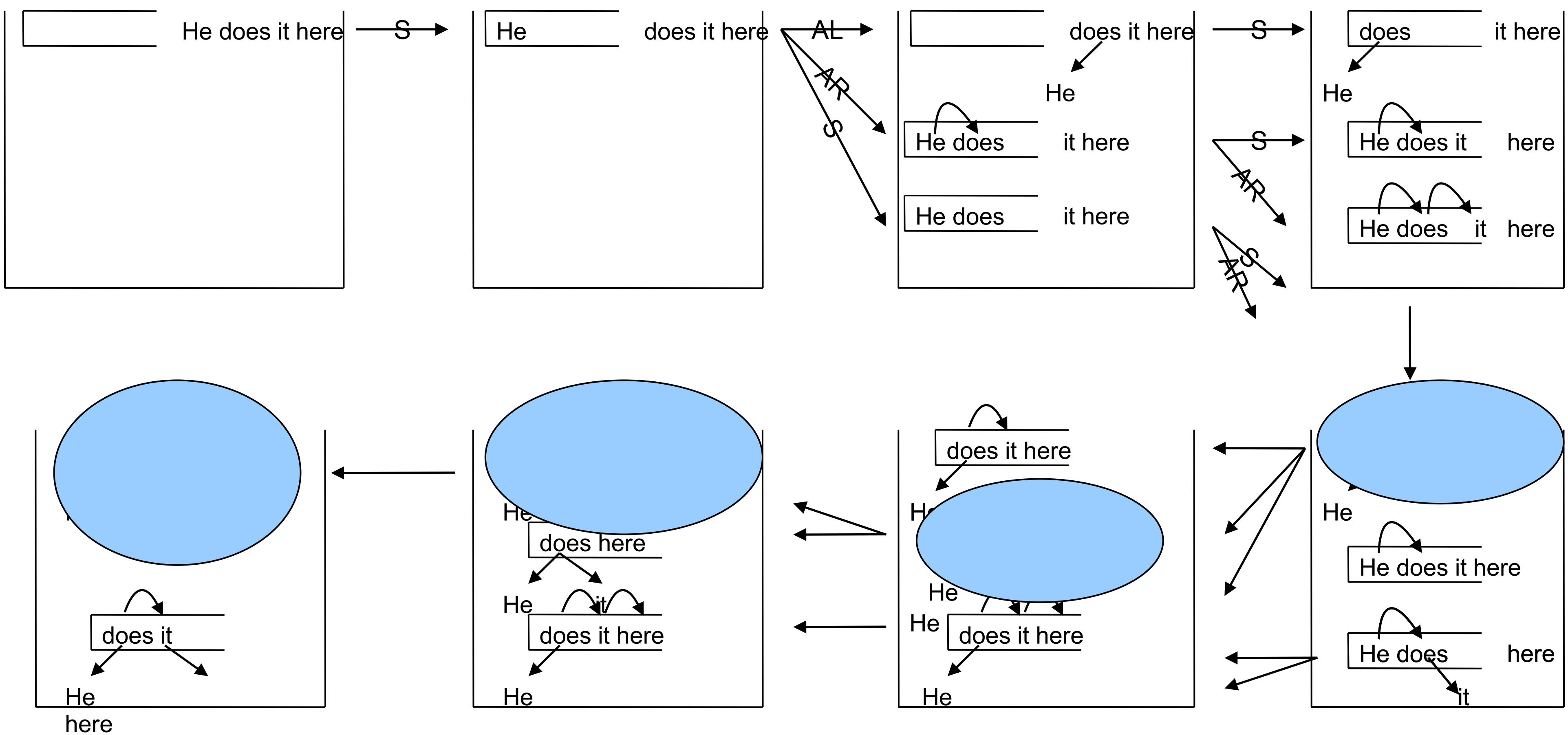
- Our parser
  - Decoding



Zhang and Clark, EMNLP 2008

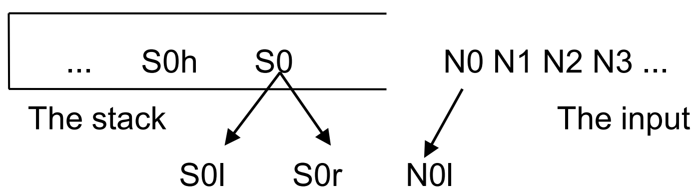
Beam-search decoding

- Our parser
  - Decoding



The feature templates

- The context



- S0 – top of stack
- S0h – head of S0
- S0l – left modifier of S0
- S0r – right modifier of S0
- N0 – head of queue
- N0l – left modifier of N0
- N1 – next in queue
- N2 – next of N1

Zhang and Clark, EMNLP 2008

The feature templates

- The base features

from single words
S0wp; S0w; S0p; N0wp; N0w; N0p;
N1wp; N1w; N1p; N2wp; N2w; N2p;

from word pairs
S0wpN0wp; S0wpN0w; S0wN0wp; S0wpN0p;
S0pN0wp; S0wN0w; S0pN0p
N0pN1p

from three words
N0pN1pN2p; S0pN0pN1p; S0hpS0pN0p;
S0pS0lpN0p; S0pS0rpN0p; S0pN0pN0lp

Zhang and Clark, EMNLP 2008

The feature templates

- The extended features
  - Distance
    - Standard in MSTParser (McDonald et al., 2005)
    - Used in easy-first (Goldberg and Elhadad, 2010)
    - When used in transition-based parsing, combined with action (this paper)

distance
S0wd; S0pd; N0wd; N0pd;
S0wN0wd; S0pN0pd;

The feature templates

- The extended features
  - Valency
    - Number of modifiers
    - Graph-based submodel of Zhang and Clark (2008)
    - The models of Martins et al. (2009)
    - The models of Sagae and Tsujii (2007)

valency
S0wvr; S0pvr; S0wvl; S0pvl; N0wvl; N0pvl;

The feature templates

- The extended features
  - Extended unigrams
    - S0h, S0l, S0r and N0l has been applied to transition-based parsers via POS-combination
    - We add their unigram word, POS and label information (this paper)

unigrams
S0hw; S0hp; S0l; S0lw; S0lp; S0ll;
S0rw; S0rp; S0rl;N0lw; N0lp; N0ll;

Zhang and Clark, ACL 2011

The feature templates

- The extended features
  - Third order
    - Graph-based dependency parsers (Carreras, 2007; Koo and Collins, 2010)

third-order
S0h2w; S0h2p; S0hl; S0l2w; S0l2p; S0l2l;
S0r2w; S0r2p; S0r2l; N0l2w; N0l2p; N0l2l;
S0pS0lpS0l2p; S0pS0rpS0r2p;
S0pS0hpS0h2p; N0pN0lpN0l2p;

The feature templates

- The extended features
  - Set of labels
    - More global feature
    - Has not been applied to transition-based parsing

label set 1
S0wsr; S0psr; S0wsl; S0psl; N0wsl; N0psl;

Experiments

- Chinese Data (CTB5)

Training, development, and test data for Chinese dependency parsing.

	Sections	Sentences	Words
Training	001–815 1,001–1,136	16,118	437,859
Dev	886–931 1,148–1,151	804	20,453
Test	816–885 1,137–1,147	1,915	50,319

- English Data (Penn Treebank)

The training, development, and test data for English dependency parsing.

	Sections	Sentences	Words
Training	2–21	39,832	950,028
Development	22	1,700	40,117
Test	23	2,416	56,684

Zhang and Clark, ACL 2011

Results

Chinese

Model	UAS	UEM	LAS
Li et al. (2012)	86.8	---	85.4
Jun et al. (2011)	86.0	35.0	---
H&S10	85.2	33.7	---
This Method	86.0	36.9	84.4

English

Model	UAS	UEM	LAS
Li et al. (2012)	93.1	---	92.0
MSTParser	91.5	42.5	
K08 standard	92.0	---	---
K&C10 model	93.0	---	---
H&S10	91.4	---	---
This Method	92.9	48.0	91.8

Zhang and Clark, ACL 2011

Applications

- Word segmentation
- Dependency parsing
- Context free grammar parsing
- Combinatory categorial grammar parsing
- Joint segmentation and POS-tagging
- Joint POS-tagging and dependency parsing
- Joint segmentation, POS-tagging and constituent parsing
- Joint segmentation, POS-tagging and dependency parsing

The shift-reduce parsing process

- We use Wang et al. (2006)'s shift-reduce transition-based process
- A state item = a pair <stack, queue>
  - Stack: holds the partial parse trees already built
  - Queue: holds the incoming words with POS
- Actions
  - SHIFT, REDUCE-BINARY-L/R, REDUCE-UNARY
  - Corresponds to arc-standard

The shift-reduce parsing process

- Actions
  - SHIFT

stack

queue

NR布朗 VV访问 NR上海

布朗(Brown) 访问(visits) 上海(Shanghai)

The shift-reduce parsing process

- Actions
  - SHIFT

stack

NR布朗

queue

VV访问 NR上海

布朗(Brown) 访问(visits) 上海(Shanghai)

Zhang and Clark, IWPT 2009

The shift-reduce parsing process

- Actions
  - REDUCE-UNARY-X

stack

NR布朗

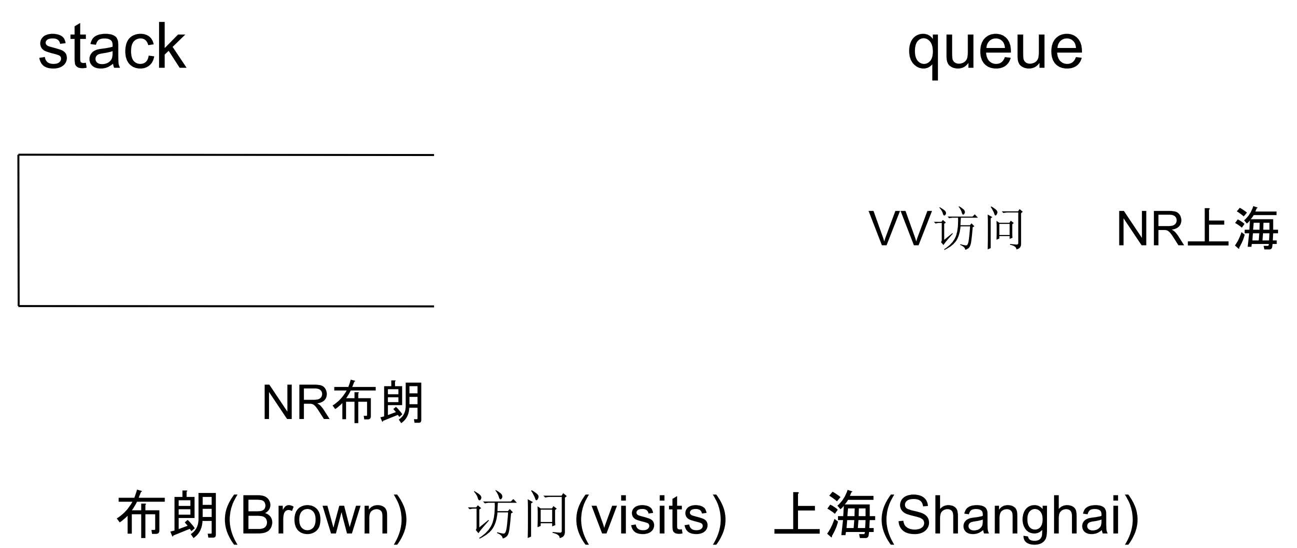
queue

VV访问 NR上海

布朗(Brown) 访问(visits) 上海(Shanghai)

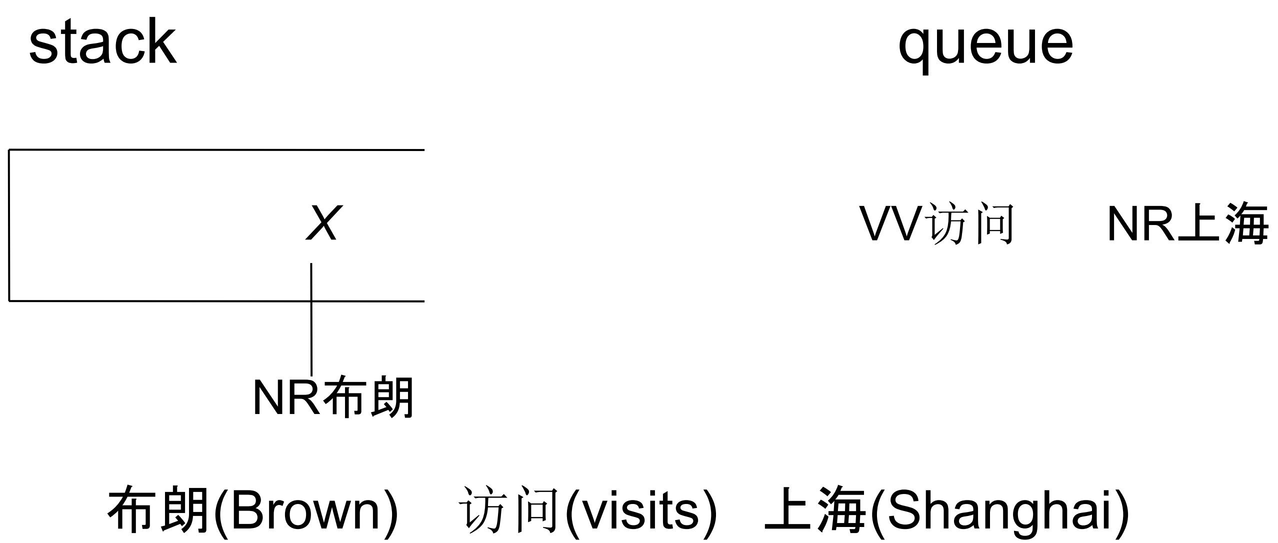
The shift-reduce parsing process

- Actions
  - REDUCE-UNARY-X



The shift-reduce parsing process

- Actions
  - REDUCE-UNARY-X

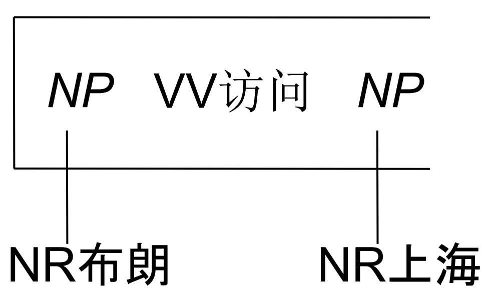


The shift-reduce parsing process

- Actions
  - REDUCE-BINARY-{L/R}-X

stack

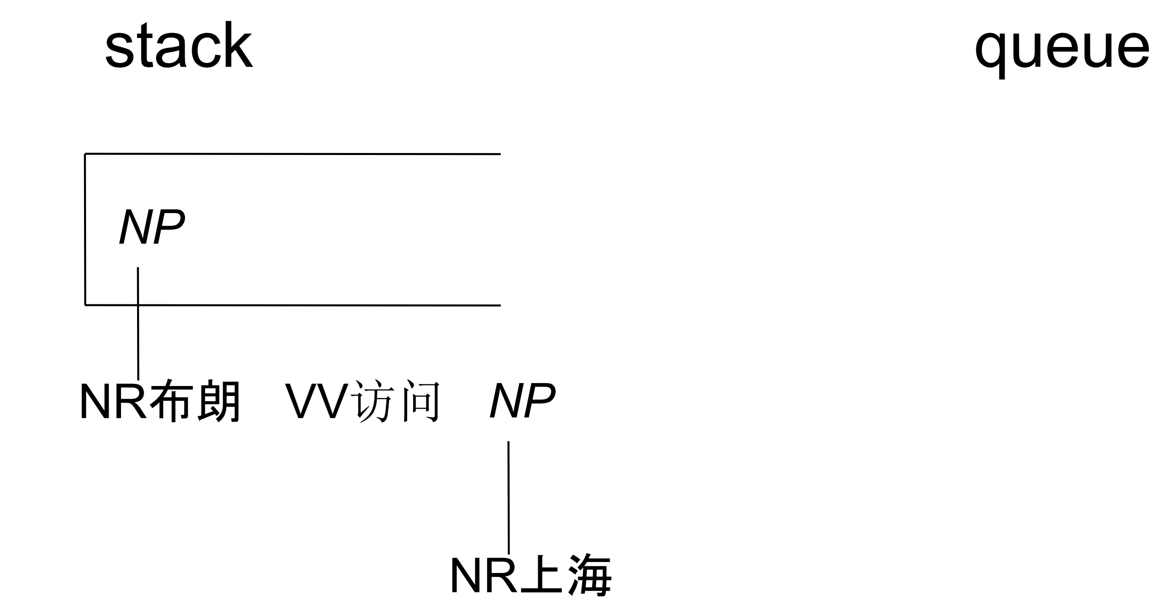
queue



布朗(Brown) 访问(visits) 上海(Shanghai)

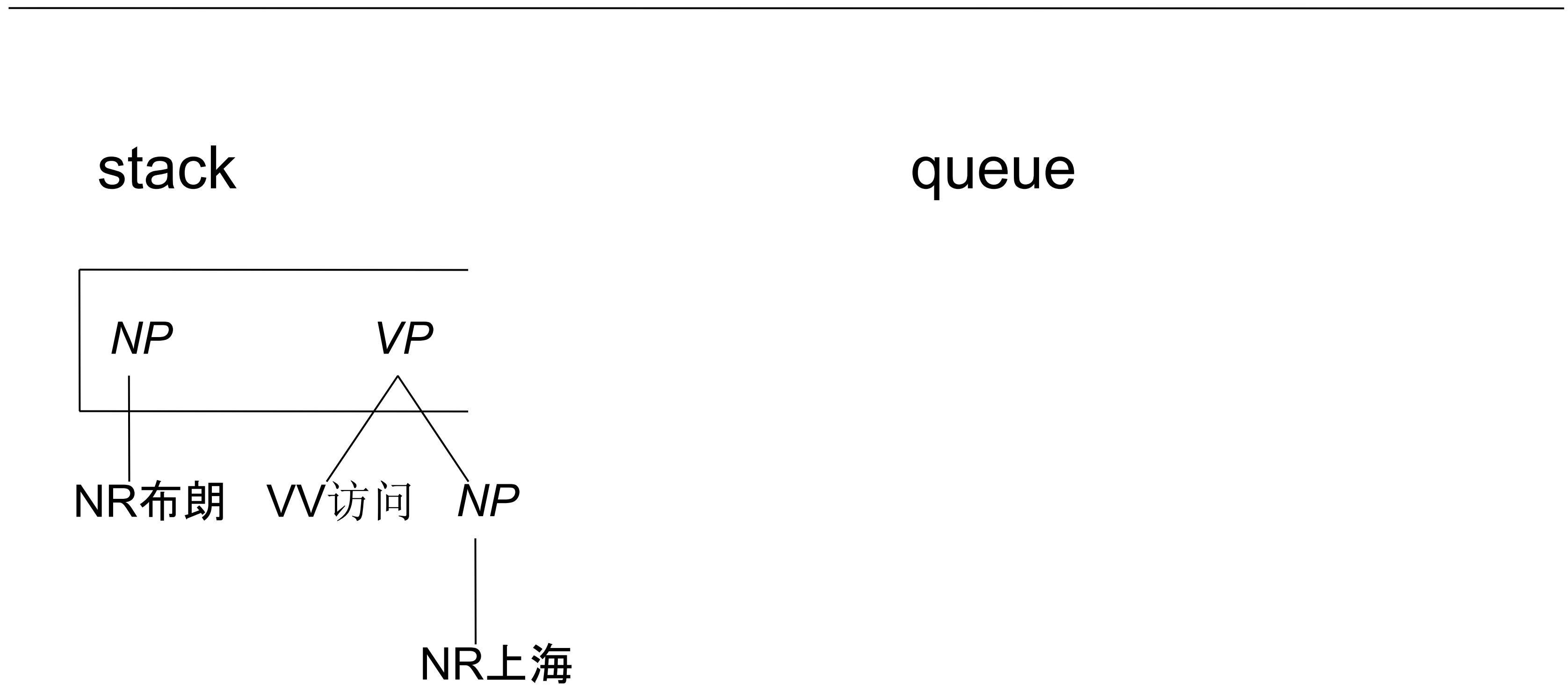
The shift-reduce parsing process

- Actions
  - REDUCE-BINARY-{L/R}-X



The shift-reduce parsing process

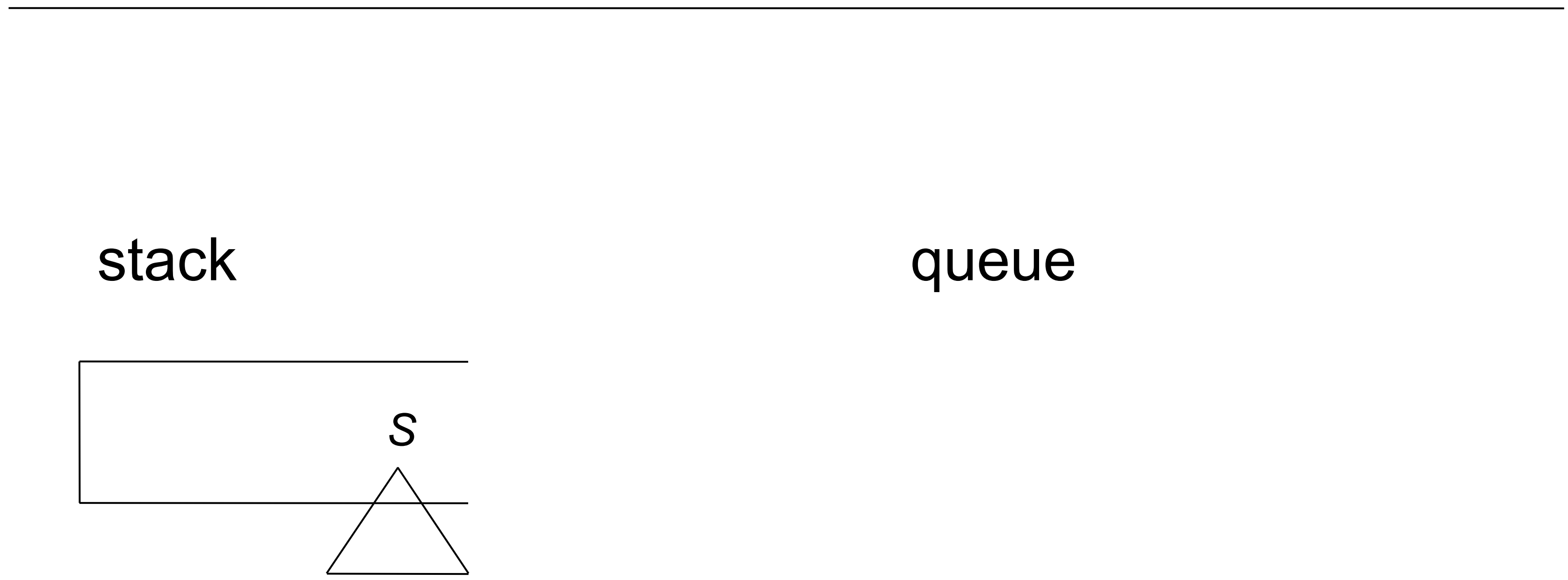
- Actions
  - REDUCE-BINARY-{L/R}-X



Zhang and Clark, IWPT 2009

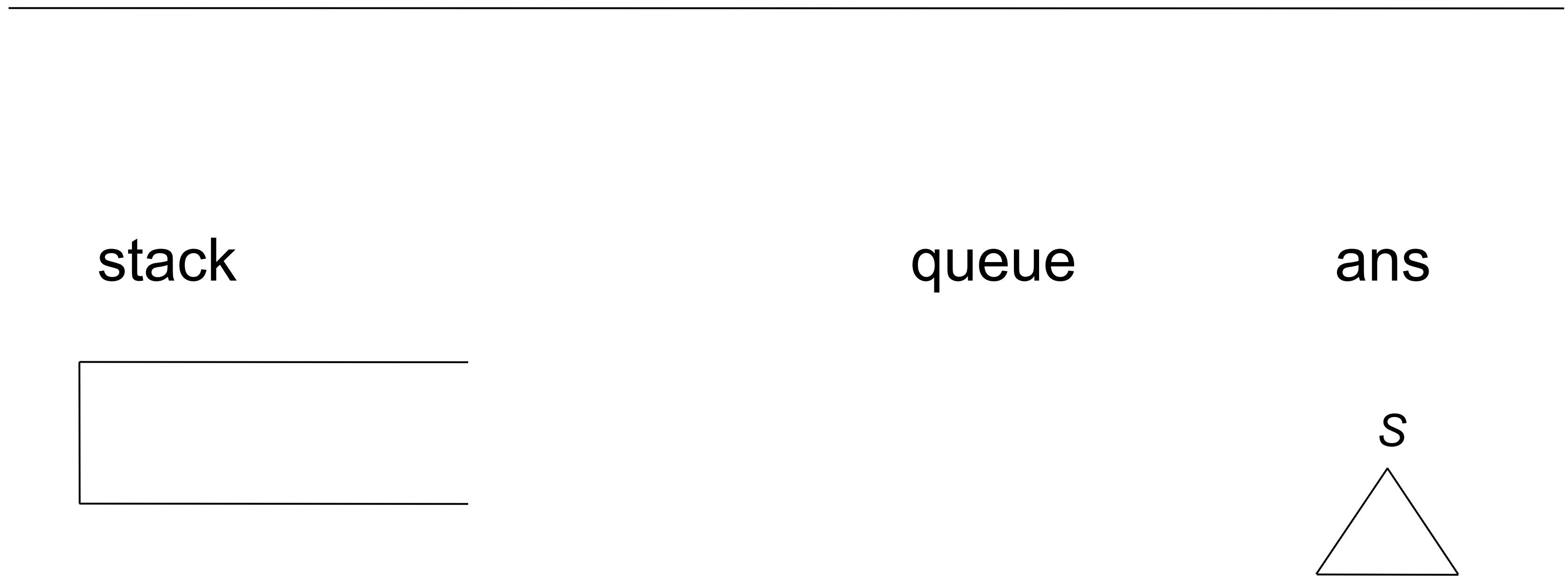
The shift-reduce parsing process

- Actions
  - TERMINATE



The shift-reduce parsing process

- Actions
  - TERMINATE



The shift-reduce parsing process

- Example
  - SHIFT

stack

queue

NR布朗 VV访问 NR上海

Zhang and Clark, IWPT 2009

The shift-reduce parsing process

- Example
  - REDUCE-UNARY-NP

stack

NR布朗

queue

VV访问 NR上海

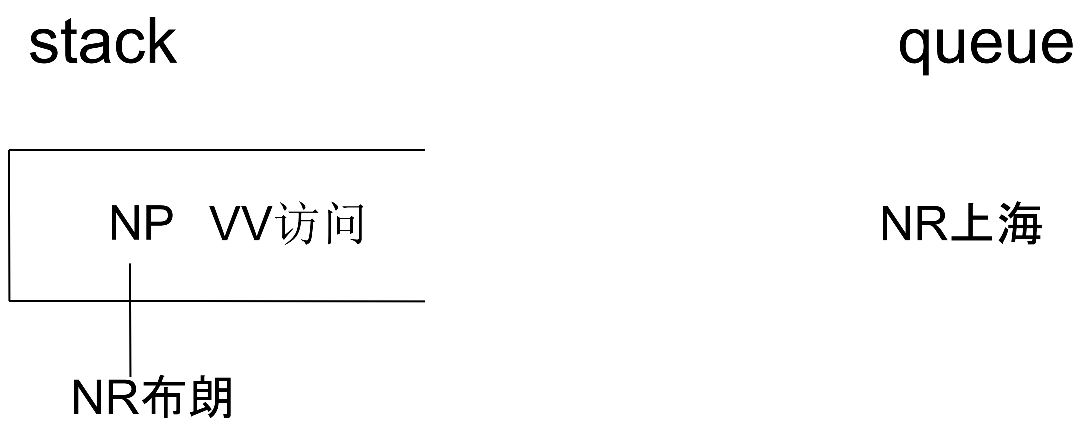
Zhang and Clark, IWPT 2009

The shift-reduce parsing process

- Example
  - SHIFT

The shift-reduce parsing process

- Example
  - SHIFT

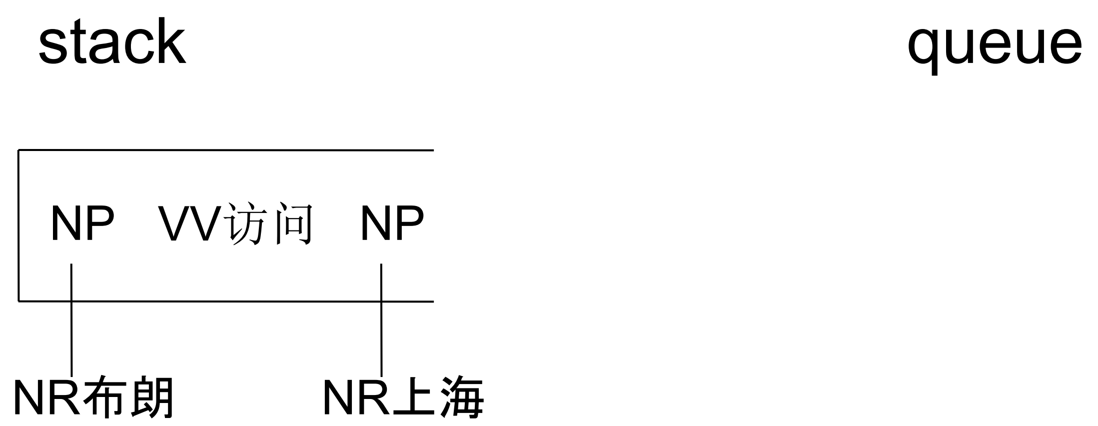


The shift-reduce parsing process

- Example
  - REDUCE-UNARY-NP

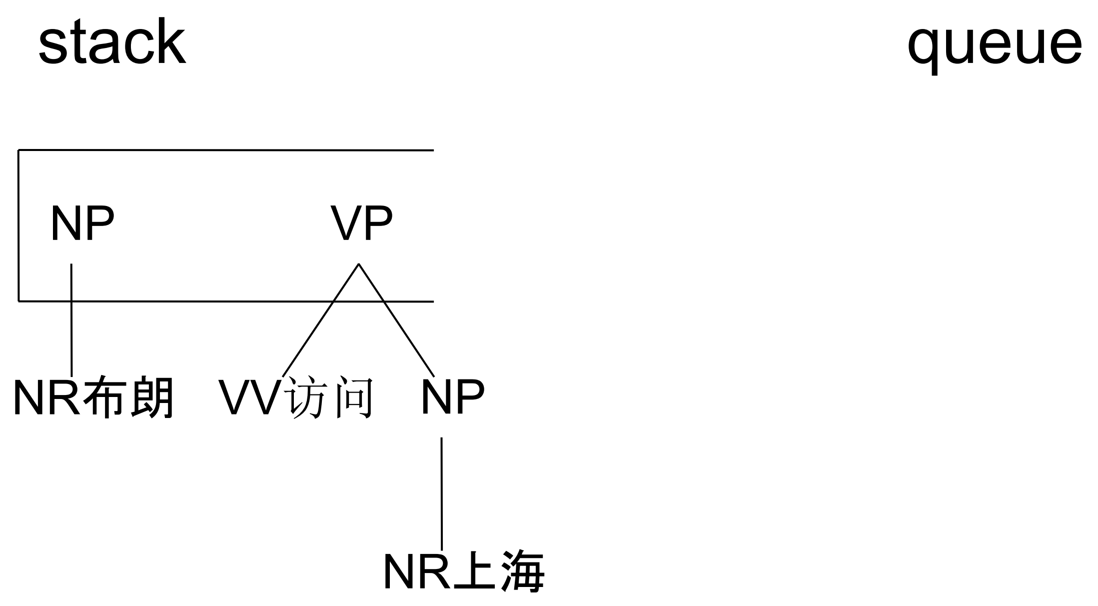
The shift-reduce parsing process

- Example
  - REDUCE-BINARY-L-VP



The shift-reduce parsing process

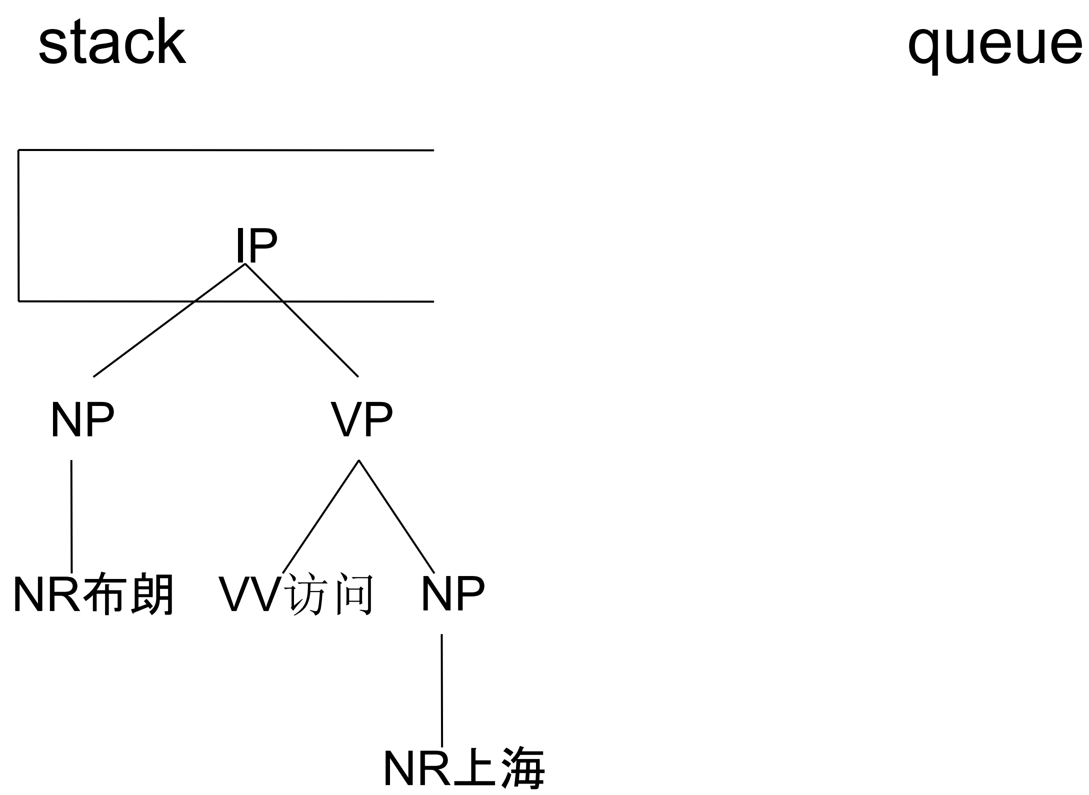
- Example
  - REDUCE-BINARY-R-IP



Zhang and Clark, IWPT 2009

The shift-reduce parsing process

- Example
  - TERMINATE

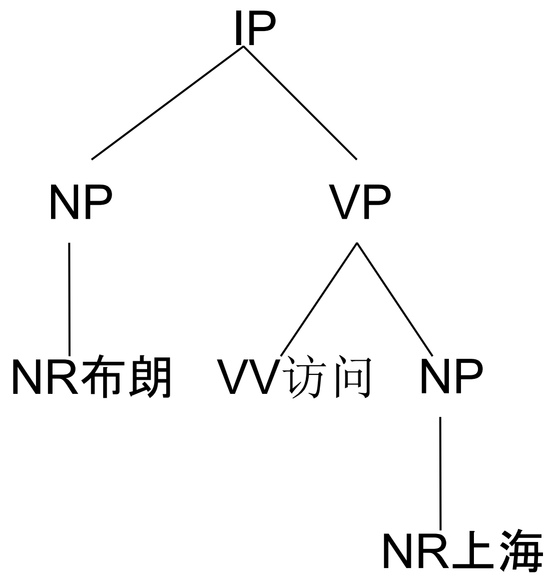


The shift-reduce parsing process

- Example

stack

queue

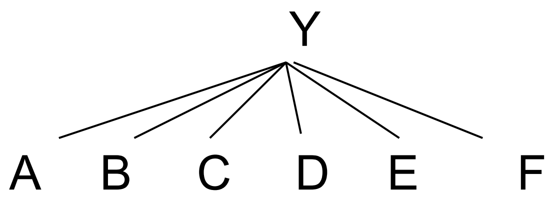


Grammar binarization

- The shift-reduce parser require binarized trees
- Treebank trees are not binarized
- Penn Treebank/CTB ↔ Parser
  - Binarize CTB data to make training data
  - Unbinarize parser output back to Treebank format
  - Reversible

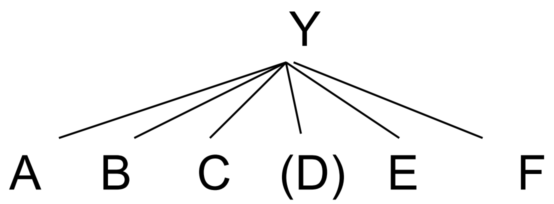
Grammar binarization

- The binarization process
  - Find head
  - Binarize left nodes
  - Binarize right nodes



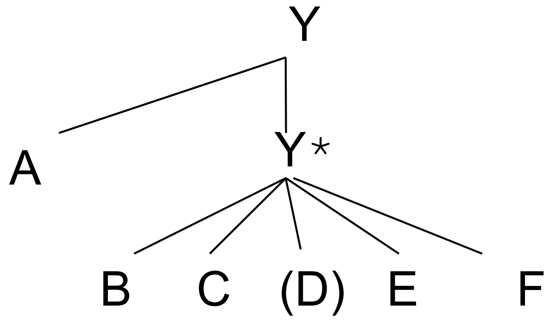
Grammar binarization

- The binarization process
  - Find head
  - Binarize left nodes
  - Binarize right nodes



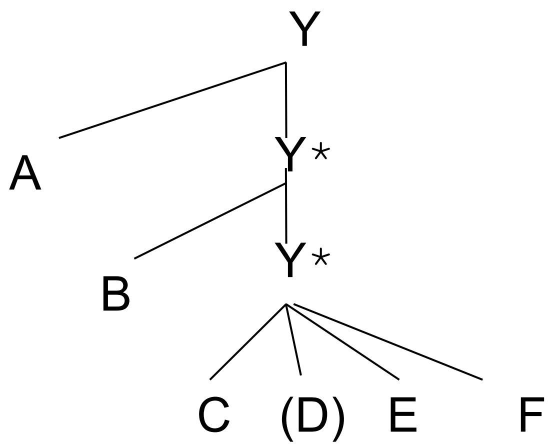
Grammar binarization

- The binarization process
  - Find head
  - Binarize left nodes
  - Binarize right nodes



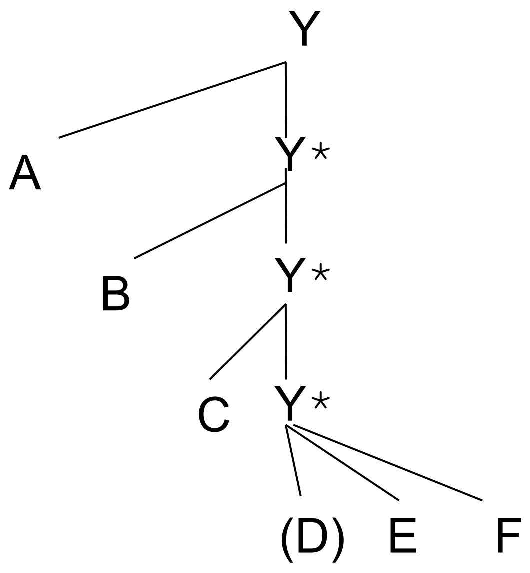
Grammar binarization

- The binarization process
  - Find head
  - Binarize left nodes
  - Binarize right nodes



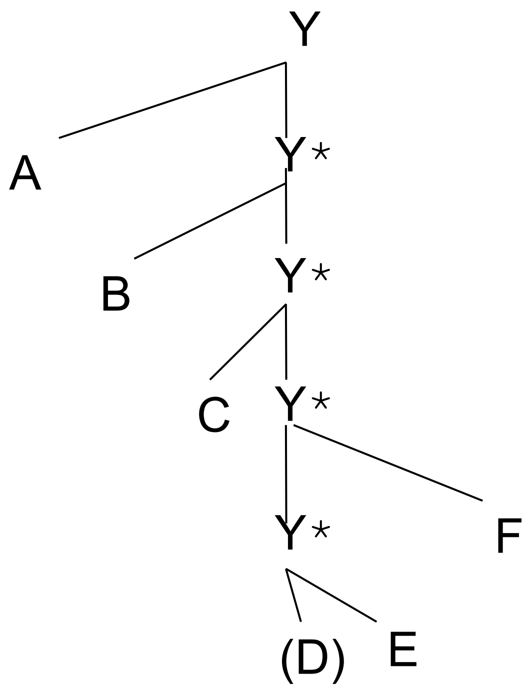
Grammar binarization

- The binarization process
  - Find head
  - Binarize left nodes
  - Binarize right nodes



Grammar binarization

- The binarization process
  - Find head
  - Binarize left nodes
  - Binarize right nodes



The statistical parser

- Beam-search decoding
  - Deterministic parsing: B=1

Initial item
stack: empty
queue: input

The statistical parser

- Beam-search decoding
  - Deterministic parsing: B=1

Initial item
stack: empty
queue: input

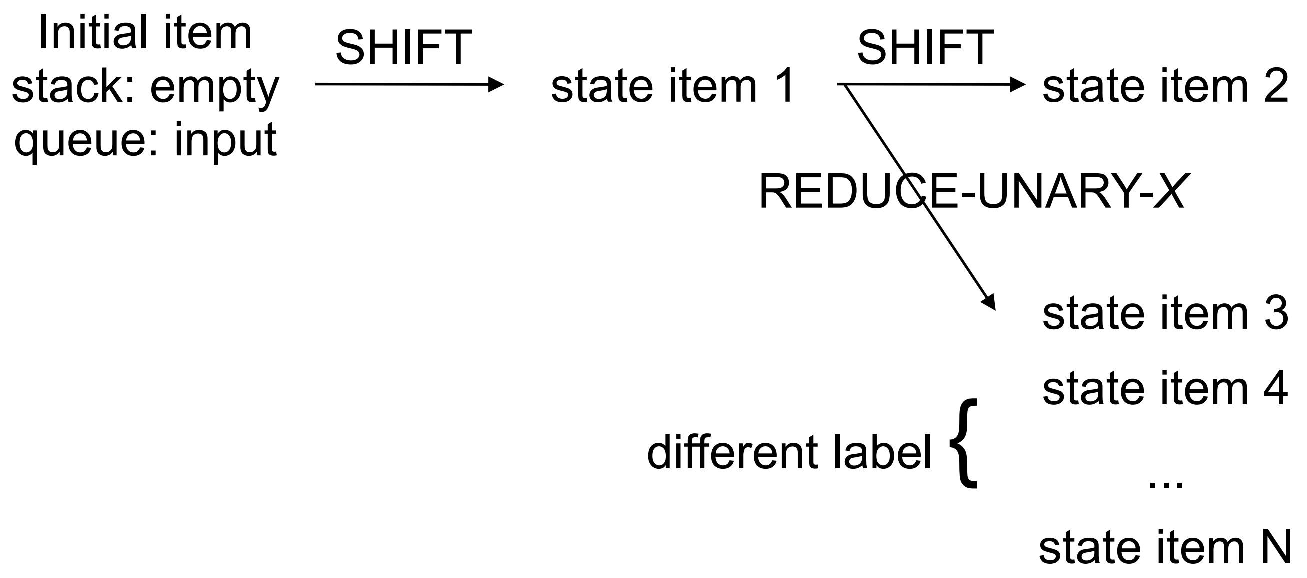
SHIFT →

state item 1

Zhang and Clark, IWPT 2009

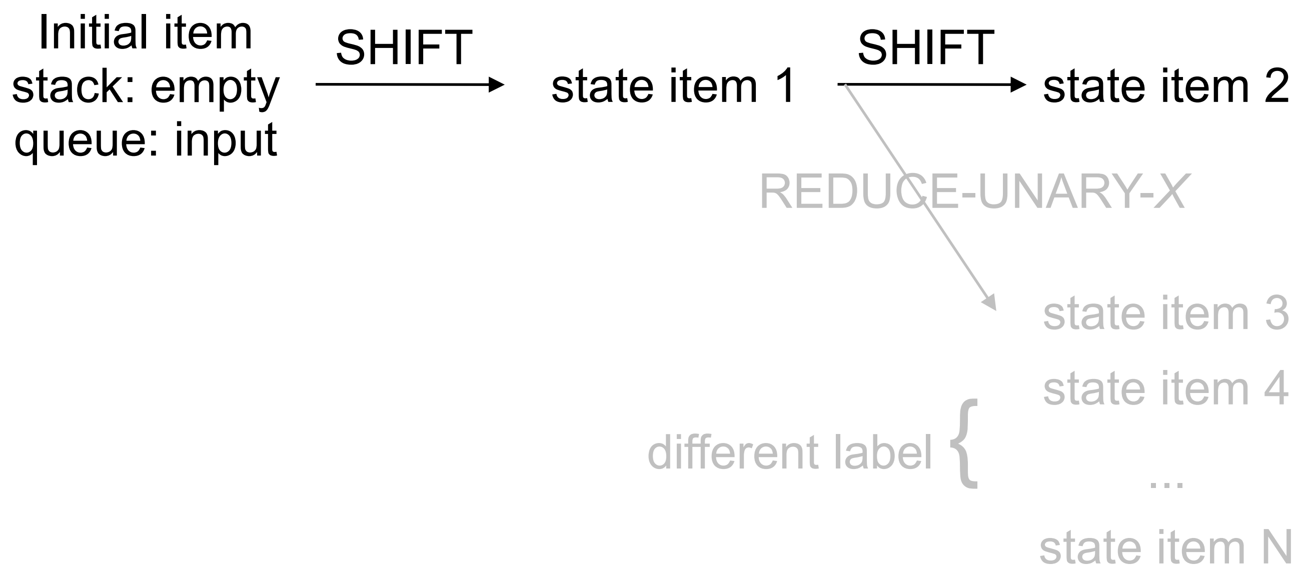
The statistical parser

- Beam-search decoding
  - Deterministic parsing: B=1



The statistical parser

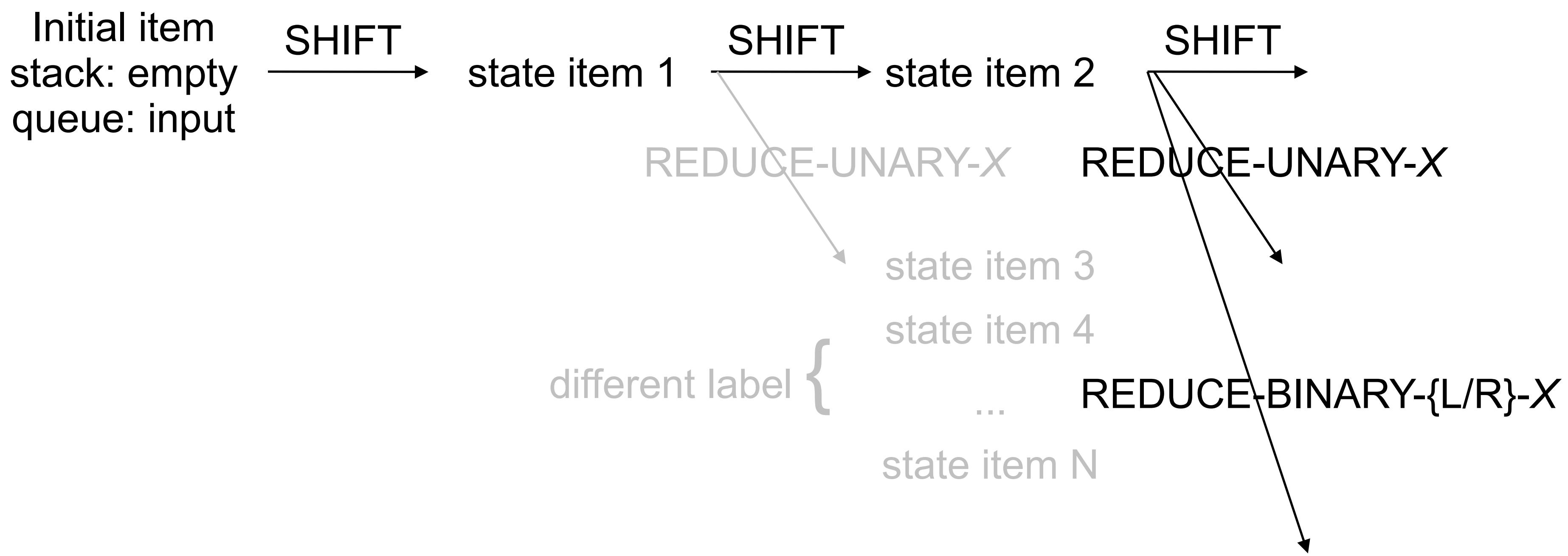
- Beam-search decoding
  - Deterministic parsing: B=1



Zhang and Clark, IWPT 2009

The statistical parser

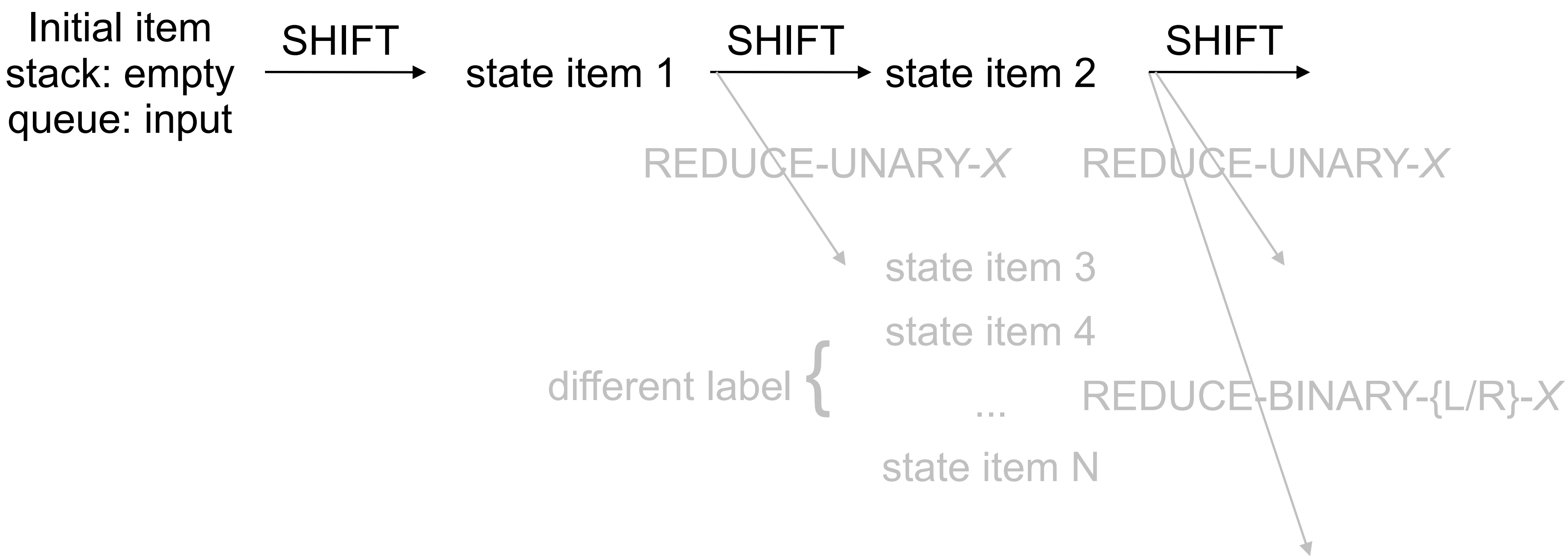
- Beam-search decoding
  - Deterministic parsing: B=1



Zhang and Clark, IWPT 2009

The statistical parser

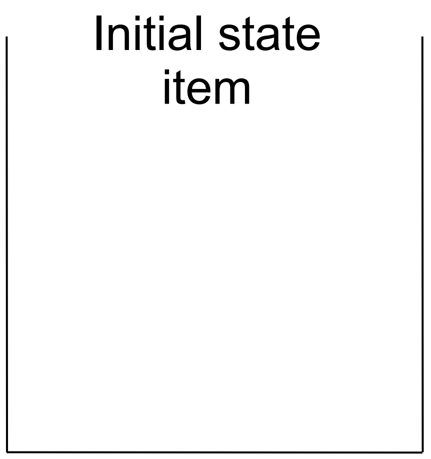
- Beam-search decoding
  - Deterministic parsing: B=1



Zhang and Clark, IWPT 2009

The statistical parser

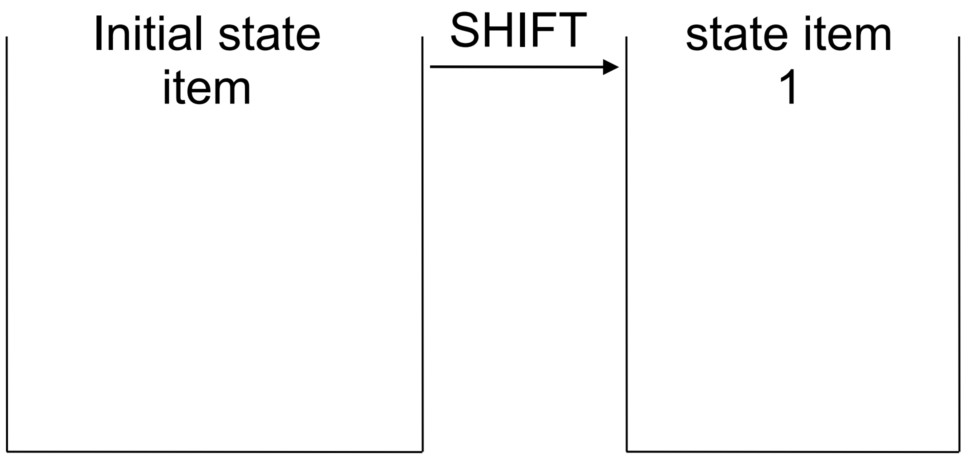
- Beam-search decoding
  - Deterministic parsing: B=1
  - Beam-search: B>1



Zhang and Clark, IWPT 2009

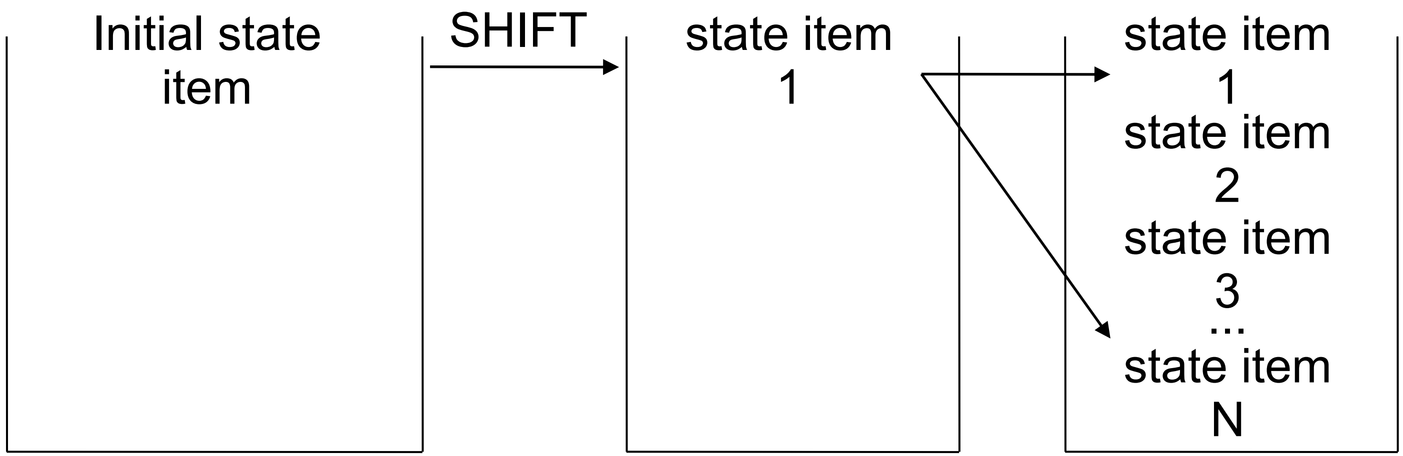
The statistical parser

- Beam-search decoding
  - Deterministic parsing: B=1
  - Beam-search: B>1



The statistical parser

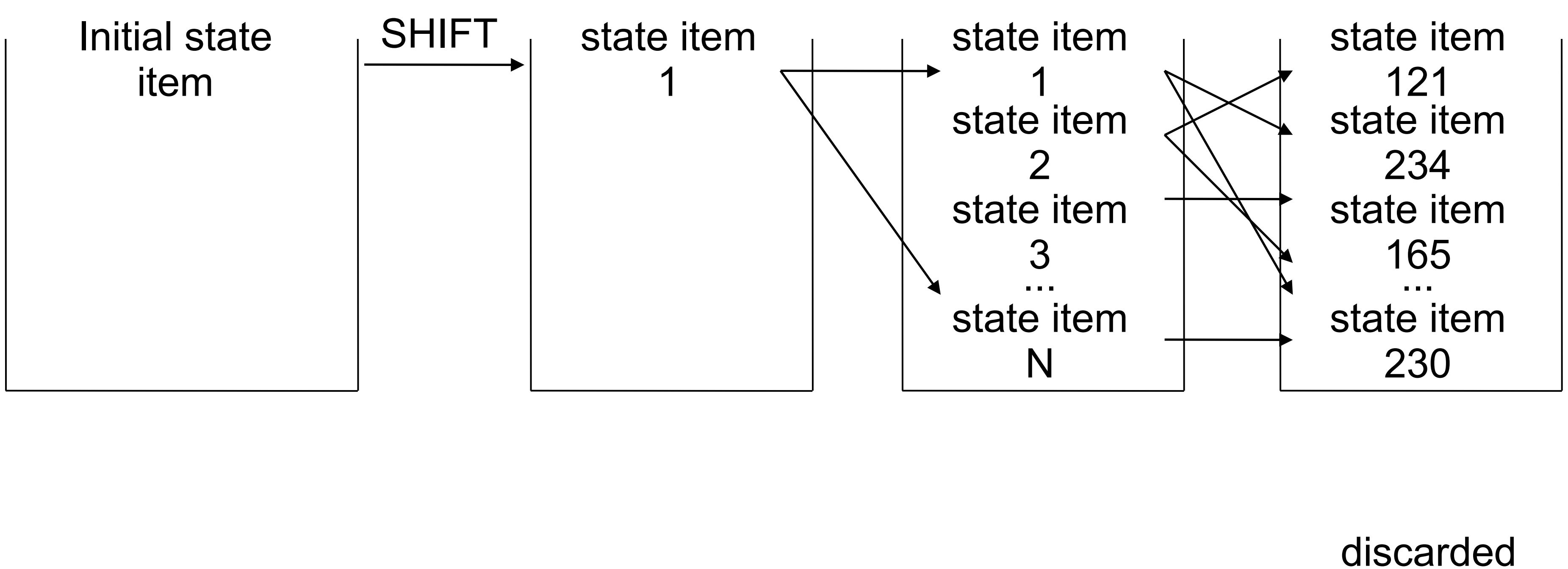
- Beam-search decoding
  - Deterministic parsing: B=1
  - Beam-search: B>1



Zhang and Clark, IWPT 2009

The statistical parser

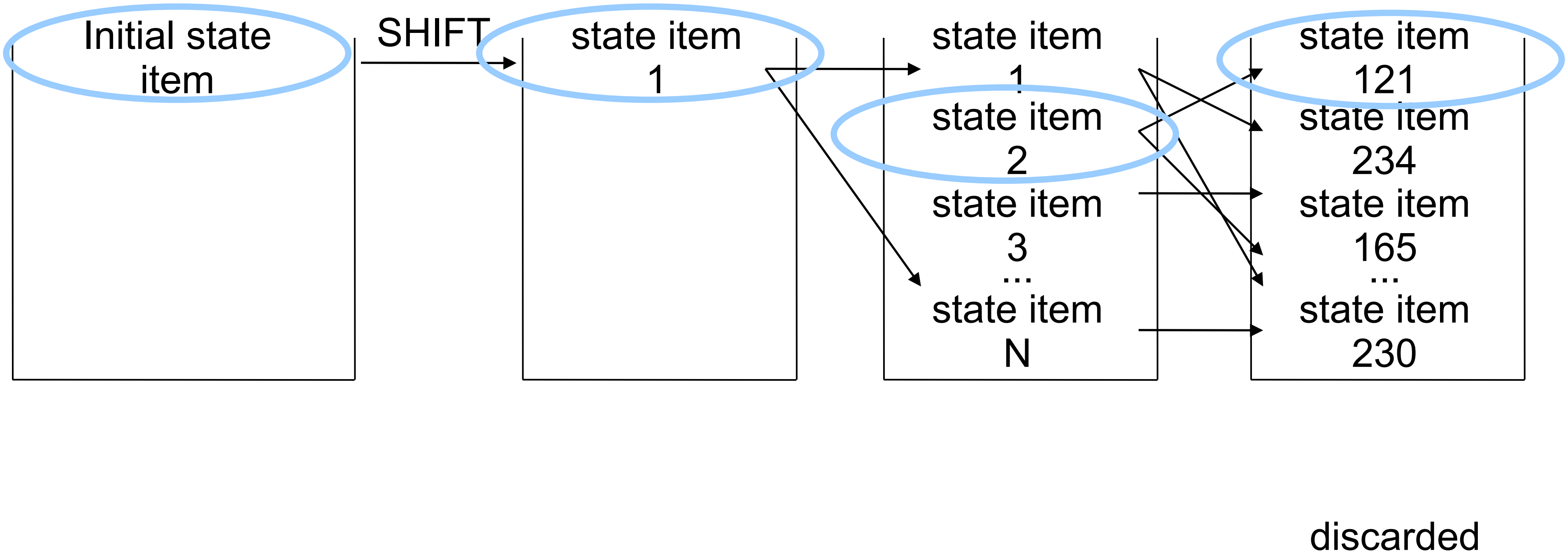
- Beam-search decoding
  - Deterministic parsing: B=1
  - Beam-search: B>1



Zhang and Clark, IWPT 2009

The statistical parser

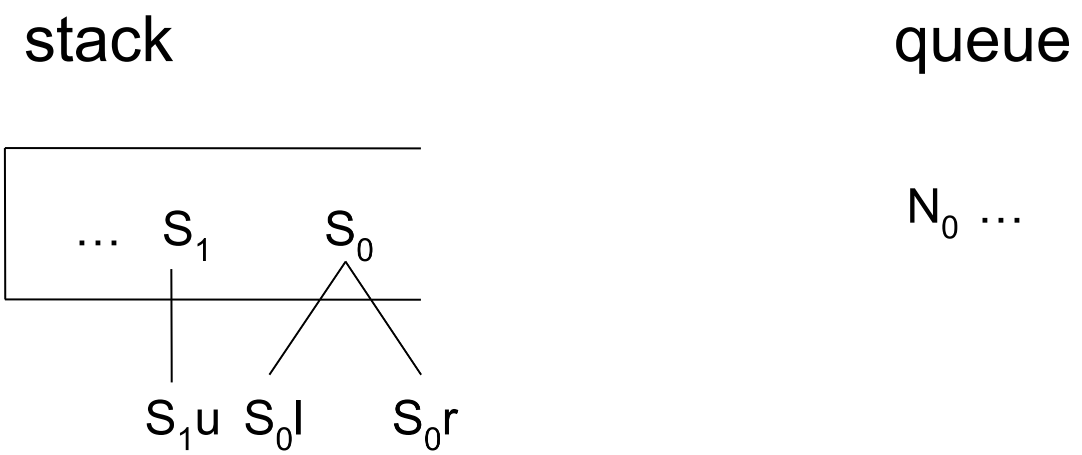
- Beam-search decoding
  - Deterministic parsing: B=1
  - Beam-search: B>1



Zhang and Clark, IWPT 2009

The statistical parser

- Features
  - Extracted from top nodes on the stack S0, S1, S2, S3, the left and right or single child of S0 and S1, and the first words on the queue N0, N1, N2, N3.



Zhang and Clark, IWPT 2009

The statistical parser

- Features
  - Manually combine word and constituent information
    - Unigrams

$$S_0tc, S_0wc, S_1tc, S_1wc,$$
$$S_2tc, S_2wc, S_3tc, S_3wc,$$
$$N_0wt, N_1wt, N_2wt, N_3wt,$$
$$S_0lwc, S_0rwc, S_0uwc,$$
$$S_1lwc, S_1rwc, S_1uwc,$$

Zhang and Clark, IWPT 2009

The statistical parser

- Features
  - Manually combine of word and constituent information
    - Bigrams

$$S_0wS_1w, S_0wS_1c, S_0cS_1w, S_0cS_1c,$$
$$S_0wN_0w, S_0wN_0t, S_0cN_0w, S_0cN_0t,$$
$$N_0wN_1w, N_0wN_1t, N_0tN_1w, N_0tN_1t$$
$$S_1wN_0w, S_1wN_0t, S_1cN_0w, S_1cN_0t,$$

Zhang and Clark, IWPT 2009

The statistical parser

- Features
  - Manually combine of word and constituent information
    - Trigrams

$$S_0cS_1cS_2c, S_0wS_1cS_2c,$$
$$S_0cS_1wS_2c, S_0cS_1cS_2w,$$
$$S_0cS_1cN_0t, S_0wS_1cN_0t,$$
$$S_0cS_1wN_0t, S_0cS_1cN_0w$$

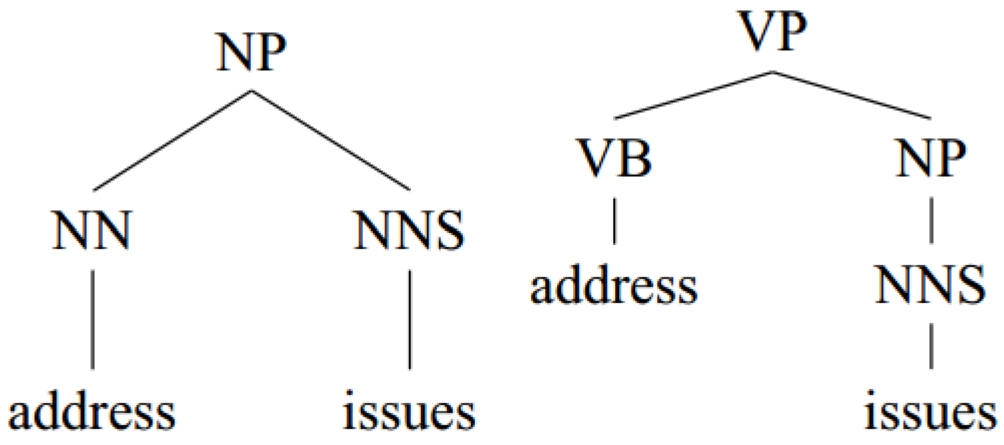
Zhang and Clark, IWPT 2009

The statistical parser

- An improvement
  - Unlike dependency parsing, different parse trees of the same input can use the different numbers of actions
  - The IDLE action
    - Align the unequal number of actions for different output trees

Zhu et al., ACL 2013

The statistical parser



LEFT: REDUCE-BINARY-R(NP), IDLE

RIGHT: REDUCE-UNARY(NP), REDUCE-BINARY-L(VP)

Zhu et al., ACL 2013

Experiments

- English PTB
- Chinese CTB51
- Standard evaluation of bracketed P, R and F

Experiments

- English results on PTB

	LR	LP	F1	#Sent/Second
Ratnaparkhi (1997)	86.3	87.5	86.9	Unk
Collins (1999)	88.1	88.3	88.2	3.5
Charniak (2000)	89.5	89.9	89.5	5.7
Sagae & Lavie (2005)	86.1	86.0	86.0	3.7
Sagae & Lavie (2006)	87.8	88.1	87.9	2.2
Petrov & Klein (2007)	90.1	90.2	90.1	6.2
Carreras et al. (2008)	90.7	91.4	91.1	Unk
This implementation	**90.2**	**90.7**	**90.4**	**89.5**

Zhu et al., ACL 2013

Experiments

- Chinese results on CTB51

	LR	LP	F1
Charniak (2000)	79.6	82.1	80.8
Bikel (2004)	79.3	82.0	80.6
Petrov & Klein (2007)	81.9	84.8	83.3
This implementation	**82.1**	**84.3**	**83.2**

Zhu et al., ACL 2013

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

Introduction to CCG parsing

- Lexical categories
  - basic categories: N (nouns), NP (noun phrases), PP (prepositional phrases), ...
  - complex categories: S\NP (intransitive verbs), (S\NP)/NP (transitive verbs), ...
- Adjacent phrases are combined to form larger phrases using category combination e.g.:
  - function application: NP S\NP $\Rightarrow$ S
  - function composition: (S\NP)/(S\NP) (S\NP)/NP $\Rightarrow$ (S\NP)/NP
- Unary rules change the type of a phrase
  - Type raising: NP $\Rightarrow$ S/(S\NP)
  - Type changing: S[pss]\NP $\Rightarrow$ NP\NP

Introduction to CCG parsing

- An example derivation

IBM bought Lotus

Introduction to CCG parsing

- An example derivation

IBM	bought	Lotus
NP	(S[dcl]\NP)/NP	NP

Introduction to CCG parsing

- An example derivation

IBM	bought	Lotus
NP	(S[dcl]\NP)/NP	NP

―――――――――――――――――――

S[dcl]\NP

Introduction to CCG parsing

- An example derivation

IBM	bought	Lotus
NP	(S[dcl]\NP)/NP	NP

S[dcl]\NP

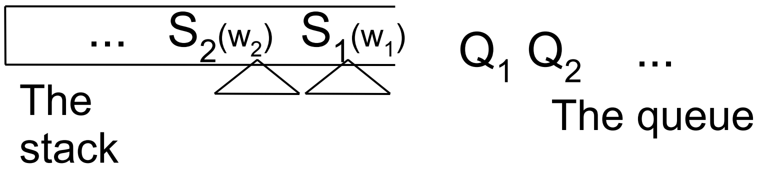
S[dcl]

Introduction to CCG parsing

- Rule extraction
  - Manually define the lexicon and combinatory rule schemas (Steedman, 2000; Clark and Curran, 2007)
  - Extracting rule instances from corpus (Hockenmaier, 2003; Fowler and Penn, 2010)

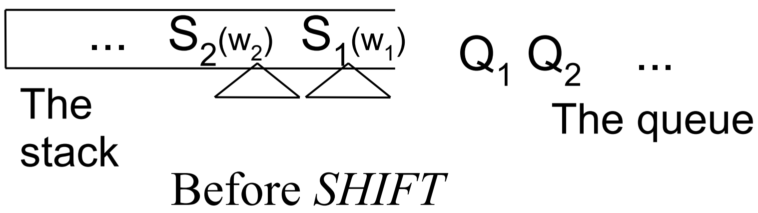
The shift-reduce parser

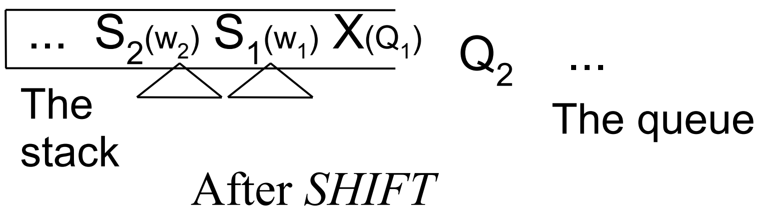
- State
  - A stack of partial derivations
  - A queue of input words
- A set of shift-reduce actions
  - SHIFT
  - COMBINE
  - UNARY
  - FINISH



The shift-reduce parser

- Shift-reduce actions
  - SHIFT-X
    - Pushes the head of the queue onto the stack
    - Assigns label X (a lexical category)
    - SHIFT action performs lexical category disambiguation

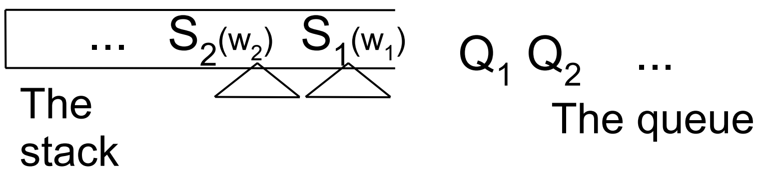




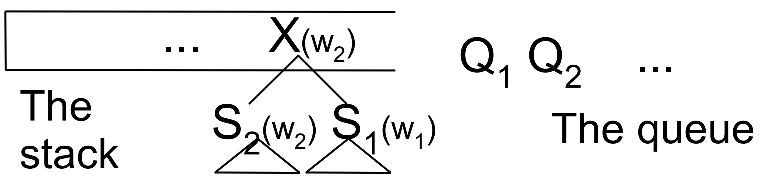
Zhang and Clark, ACL 2011

The shift-reduce parser

- Shift-reduce actions
  - COMBINE-X
    - Pops the top two nodes off the stack
    - Combines into a new node X, and push it onto stack
    - Corresponds to the use of a combinatory rule in CCG



Before *COMBINE*

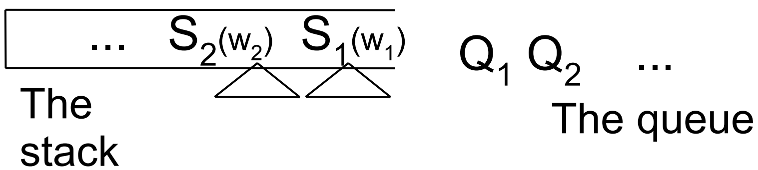


After *COMBINE*

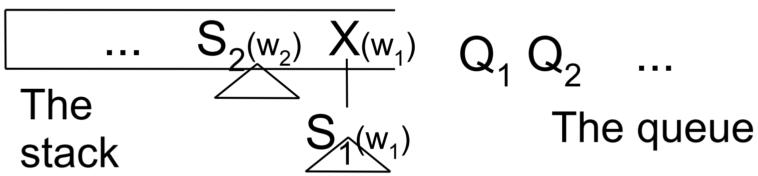
Zhang and Clark, ACL 2011

The shift-reduce parser

- Shift-reduce actions
  - UNARY-X
    - Pops the top of the stack
    - Create a new node with category X; pushes it onto stack
    - Corresponds to the use of a unary rule in CCG



Before *UNARY*



After *UNARY*

Zhang and Clark, ACL 2011

The shift-reduce parser

- Shift-reduce actions
  - FINISH
    - Terminates the parsing process
    - Can be applied when all input words have been pushed onto the stack
    - Allows fragmentary analysis:
      - when the stack holds multiple items that cannot be combined
      - such cases can arise from incorrect lexical category assignment

The shift-reduce parser

- An example parsing process

IBM bought Lotus yesterday

initial

The shift-reduce parser

- An example parsing process

NP_{IBM} bought Lotus yesterday

SHIFT

The shift-reduce parser

- An example parsing process

Stack	Input
NP_{IBM} $((S[dcl]\backslash NP)/NP)_{bought}$	Lotus yesterday

SHIFT

Zhang and Clark, ACL 2011

The shift-reduce parser

- An example parsing process

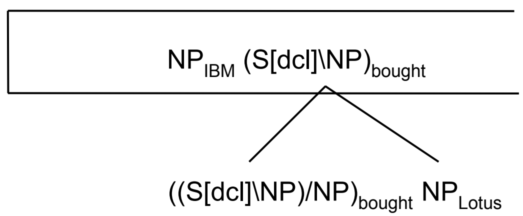
NP_{IBM} $((S[dcl]\backslash NP)/NP)_{bought}$ NP_{Lotus}

yesterday

SHIFT

The shift-reduce parser

- An example parsing process



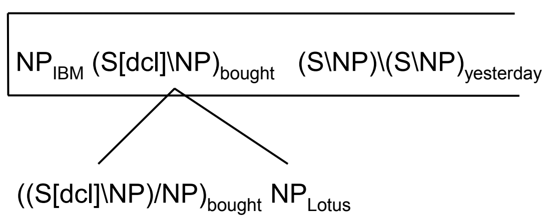
yesterday

COMBINE

Zhang and Clark, ACL 2011

The shift-reduce parser

- An example parsing process

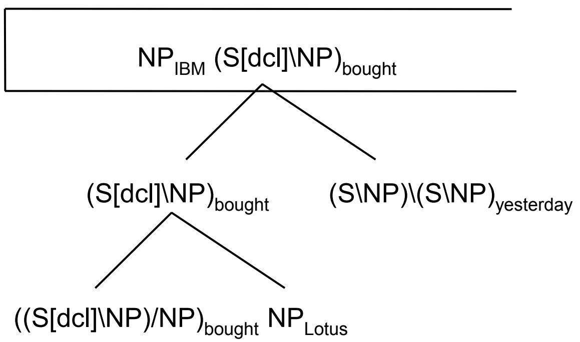


SHIFT

Zhang and Clark, ACL 2011

The shift-reduce parser

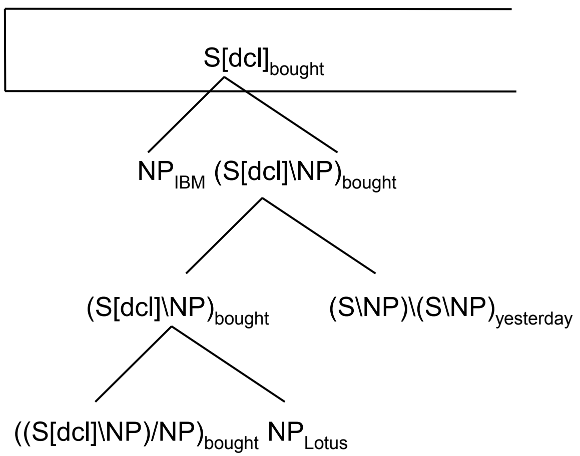
- An example parsing process



COMBINE

The shift-reduce parser

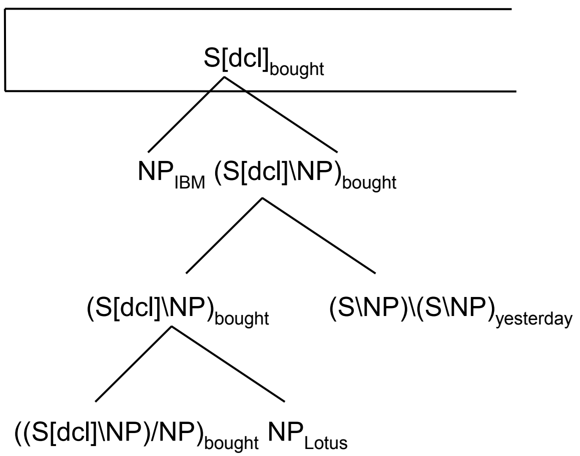
- An example parsing process



COMBINE

The shift-reduce parser

- An example parsing process

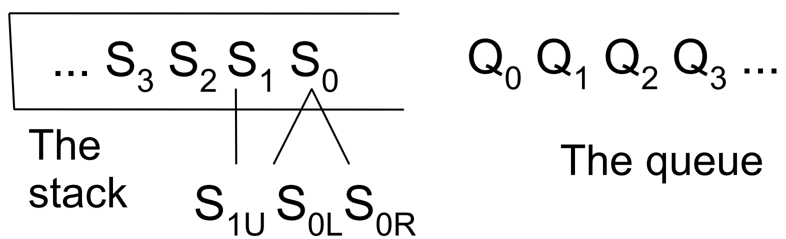


FINISH

Zhang and Clark, ACL 2011

Features

- Beam-search decoding
  - context



  - Stack nodes: S0 S1 S2 S3
  - Queue nodes: Q0 Q1 Q2 Q3
  - Stack subnodes: S0L S0R S0U S1L/R/U

S0wp, S0c, S0pc, S0wc, S1wp, S1c, S1pc, S1wc, S2pc, S2wc, S3pc, S3wc,
Q0wp, Q1wp, Q2wp, Q3wp,
S0Lpc, S0Lwc, S0Rpc, S0Rwc, S0Upc, S0Uwc, S1Lpc, S1Lwc, S1Rpc, S1Rwc, S1Upc, S1Uwc,
S0wcS1wc, S0cS1w, S0wS1c, S0cS1c, S0wcQ0wp, S0cQ0wp, S0wcQ0p, S0cQ0p, S1wcQ0wp, S1cQ0wp, S1wcQ0p, S1cQ0p,
S0wcS1cQ0p, S0cS1wcQ0p, S0cS1cQ0wp, S0cS1cQ0p, S0pS1pQ0p, S0wcQ0pQ1p, S0cQ0wpQ1p, S0cQ0pQ1wp, S0cQ0pQ1p, S0pQ0pQ1p, S0wcS1cS2c, S0cS1wcS2c, S0cS1cS2wc, S0cS1cS2c, S0pS1pS2p,
S0cS0HcS0Lc, S0cS0HcS0Rc, S1cS1HcS1Rc, S0cS0RcQ0p, S0cS0RcQ0w, S0cS0LcS1c, S0cS0LcS1w, S0cS1cS1Rc, S0wS1cS1Rc.

Zhang and Clark, ACL 2011

Experimental data

- CCGBank (Hockenmaier and Steedman, 2007)
- Split into three subsets:
  - Training (section 02 – 21)
  - Development (section 00)
  - Testing (section 23)
- Extract CCG rules
  - Binary instances: 3070
  - Unary instances: 191
- Evaluation F-score over CCG dependencies
  - Use C&C tools for transformation

Test results

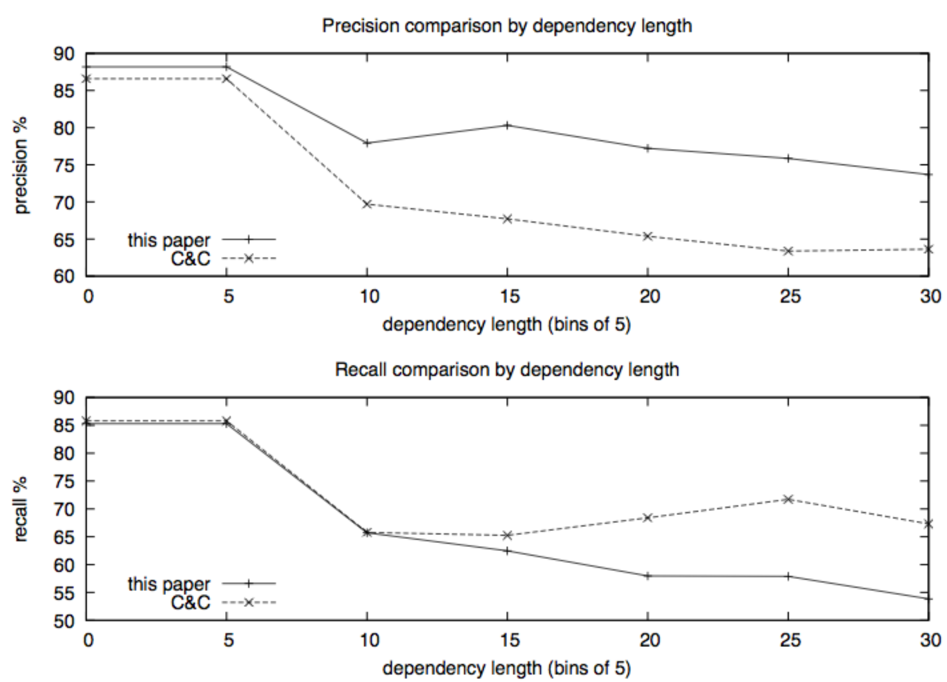
- F&P = Fowler and Penn (2010)

	LP	LR	LF	lsent.	cats.	evaluated
shift-reduce	**87.43**	**83.61**	**85.48**	**35.19**	**93.12**	**all sentences**
C&C (normal-form)	85.58	82.85	84.20	32.90	92.84	all sentences
shift-reduce	87.43	83.71	85.53	35.34	93.15	99.58% (C&C coverage)
C&C (hybrid)	86.17	84.74	85.45	32.92	92.98	99.58% (C&C coverage)
C&C (normal-form)	85.48	84.60	85.04	33.08	92.86	99.58% (C&C coverage)
F&P (Petrov I-5)*	86.29	85.73	86.01	--	--	-- (F&P ∩ C&C coverage; 96.65% on dev. test)
C&C hybrid*	86.46	85.11	85.78	--	--	-- (F&P ∩ C&C coverage; 96.65% on dev. test)

Zhang and Clark, ACL 2011

Error Comparisons

- As sentence length increases
  - Both parsers give lower performance
  - No difference in the rate of accuracy degradation
- When dependency length increases



Zhang and Clark, ACL 2011

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

Introduction of Chinese POS-tagging

- Word segmentation is a necessary step before POS-tagging

Input	我喜欢读书	Ilikereadingbooks
Segment	我 喜欢 读 书	I like reading books
Tag	我/PN 喜欢/V 读/V 书/N	I/PN like/V reading/V books/N

- The traditional approach treats word segmentation and POS-tagging as two separate steps

Two observations

- Segmentation errors propagate to the step of POS-tagging

Input	我喜欢读书	Ilikereadingbooks
Segment	**我喜** **欢** 读 书	**Ili** **ke** reading books
Tag	**我喜/N** **欢/V** 读/V 书/N	**Ili/N** **ke/V** reading/V books/N

- Information about POS helps to improve segmentation

一/CD (1) 个/M (measure word) 人/N (person) or 一/CD (1) 个人/JJ (personal)
二百三十三/CD (233) or 二/CD (2) 百/CD (hundred) 三/CD (3) 十/CD (ten) 三/CD (3)

Joint segmentation and tagging

- The observations lead to the solution of joint segmentation and POS-tagging

  Input 我喜欢读书 Ilikereading

  Output 我/PN 喜欢/V 读/V 书/N I/PN like/V reading/V books/N

- Consider segmentation and POS information simultaneously
- The most appropriate output is chosen from all possible segmented and tagged outputs

The transition system

- State
  - Partial segmented results
  - Unprocessed characters
- Two actions
  - Separate (t) : t is a POS tag
  - Append

The transition system

- Initial state

我喜欢读书

Zhang and Clark, EMNLP 2010

The transition system

- Separate(PN)

我/PN

喜欢读书

The transition system

- Separate (V)

我/PN　喜/V

欢读书

The transition system

- Append

我/PN　喜欢/V

读书

The transition system

- Separate (V)

我/PN　喜欢/V　读/V

书

The transition system

- Separate (N)

我/PN 喜欢/V 读/V 书/N

The transition system

- End state

我/PN　喜欢/V　读/V　书/N

Feature templates

Feature templates for the word segmentor.

	Feature template	When c_0 is
1	w_{-1}	separated
2	$w_{-1}w_{-2}$	separated
3	w_{-1}, where $len(w_{-1}) = 1$	separated
4	$start(w_{-1})len(w_{-1})$	separated
5	$end(w_{-1})len(w_{-1})$	separated
6	$end(w_{-1})c_0$	separated
7	$c_{-1}c_0$	appended
8	$begin(w_{-1})end(w_{-1})$	separated
9	$w_{-1}c_0$	separated
10	$end(w_{-2})w_{-1}$	separated
11	$start(w_{-1})c_0$	separated
12	$end(w_{-2})end(w_{-1})$	separated
13	$w_{-2}len(w_{-1})$	separated
14	$len(w_{-2})w_{-1}$	separated

w = word; c = character. The index of the current character is 0.

Zhang and Clark, EMNLP 2010

Feature templates

POS feature templates for the joint segmentor and POS-tagger.

	Feature template	when c_0 is
1	$w_{-1}t_{-1}$	separated
2	$t_{-1}t_0$	separated
3	$t_{-2}t_{-1}t_0$	separated
4	$w_{-1}t_0$	separated
5	$t_{-2}w_{-1}$	separated
6	$w_{-1}t_{-1}end(w_{-2})$	separated
7	$w_{-1}t_{-1}c_0$	separated
8	$c_{-2}c_{-1}c_0t_{-1}$, where $len(w_{-1}) = 1$	separated
9	c_0t_0	separated
10	$t_{-1}start(w_{-1})$	separated
11	t_0c_0	separated or appended
12	$c_0t_0start(w_0)$	appended
13	$ct_{-1}end(w_{-1})$, where $c \in w_{-1}$ and $c \neq end(w_{-1})$	separated
14	$c_0t_0cat(start(w_0))$	separated
15	$ct_{-1}cat(end(w_{-1}))$, where $c \in w_{-1}$ and $c \neq end(w_{-1})$	appended
16	$c_0t_0c_{-1}t_{-1}$	separated
17	$c_0t_0c_{-1}$	appended

w = word; c = character; t = POS-tag. The index of the current character is 0.

Zhang and Clark, EMNLP 2010

Experiments

- Penn Chinese Treebank 5 (CTB-5)

	CTB files	# sent.	# words
Training	1-270	18089	493,939
	400-1151		
Develop	301-325	350	6,821
Test	271-300	348	8,008

Experiments

Accuracy comparisons between various joint segmentors and POS-taggers on CTB5

	SF	JF
K09 (error-driven)	97.87	93.67
This work	**97.78**	**93.67**
Zhang 2008	97.82	93.62
K09 (baseline)	97.79	93.60
J08a	97.85	93.41
J08b	97.74	93.37
N07	97.83	93.32

SF = segmentation F-score; JF = joint segmentation and POS-tagging F-score

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

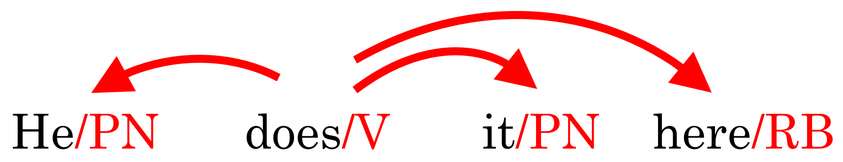
Introduction

- Traditional dependency parsing
  - Input: POS-tagged sentence e.g He/PN does/V it/PN here/RB
  - Output:

- Accurate dependency parsing heavily relies on POS tagging information
- Error propagation
- Syntactic information can be helpful for POS disambiguation

Introduction

- Joint POS-tagging and dependency parsing
  - Input: POS-tagged sentence e.g He does it here
  - Output:



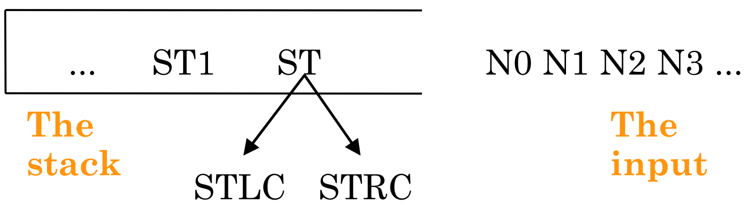
The extended arc-standard transition system

- Extended arc-standard dependency parsing transition
- State
  - A stack to hold partial candidates
  - A queue of next incoming words
- Four actions
  - SHIFT(t), LEFT-REDUCE, RIGHT-REDUCE
    t is the POS tag

Hatori et al. IJCNLP 2011

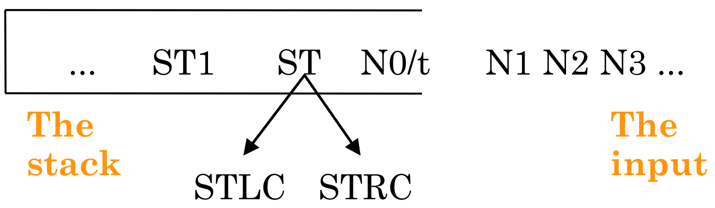
The extended arc-standard transition system

- Actions
  - SHIFT(t)



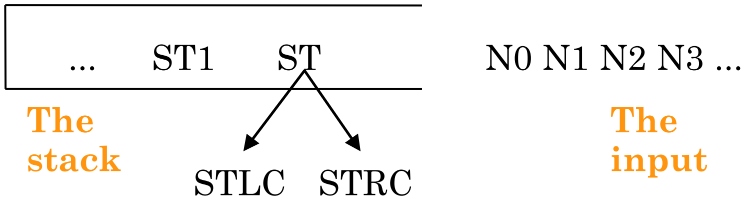
The extended arc-standard transition system

- Actions
  - SHIFT(t)
    - Pushes stack



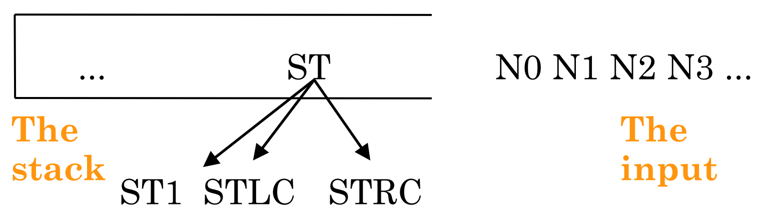
The extended arc-standard transition system

- Actions
  - LEFT-REDUCE



The extended arc-standard transition system

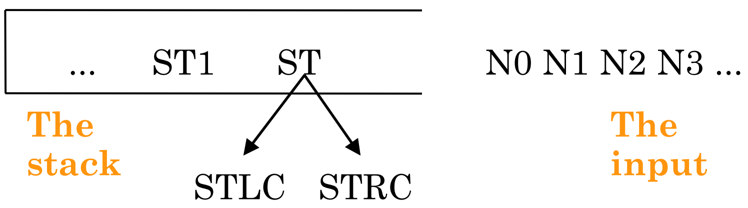
- Actions
  - LEFT-REDUCE
    - Pops stack
    - Adds link



Hatori et al. IJCNLP 2011

The extended arc-standard transition system

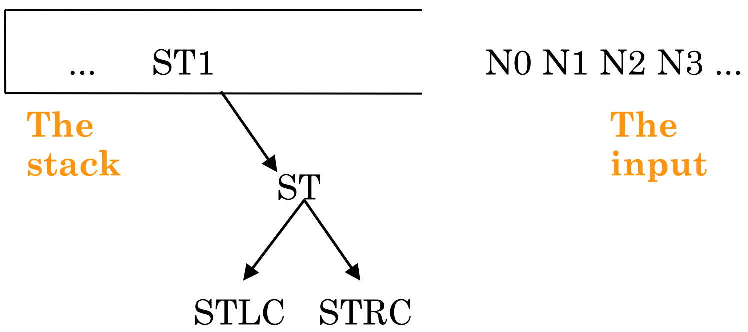
- Actions
  - RIGHT-REDUCE



Hatori et al. IJCNLP 2011

The extended arc-standard transition system

- Actions
  - RIGHT-REDUCE
    - Pops stack
    - Adds link



Hatori et al. IJCNLP 2011

The extended arc-standard transition system

- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE

He does it here

Hatori et al. IJCNLP 2011

The extended arc-standard transition system

- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE

Hatori et al. IJCNLP 2011

The extended arc-standard transition system

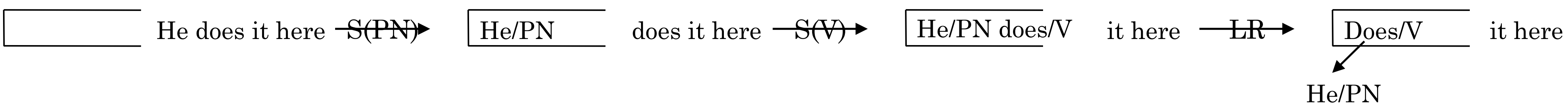
- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE

[ ] He does it here —S(PN)→ [He/PN] does it here —S(V)→ [He/PN does/V] it here

Hatori et al. IJCNLP 2011

The extended arc-standard transition system

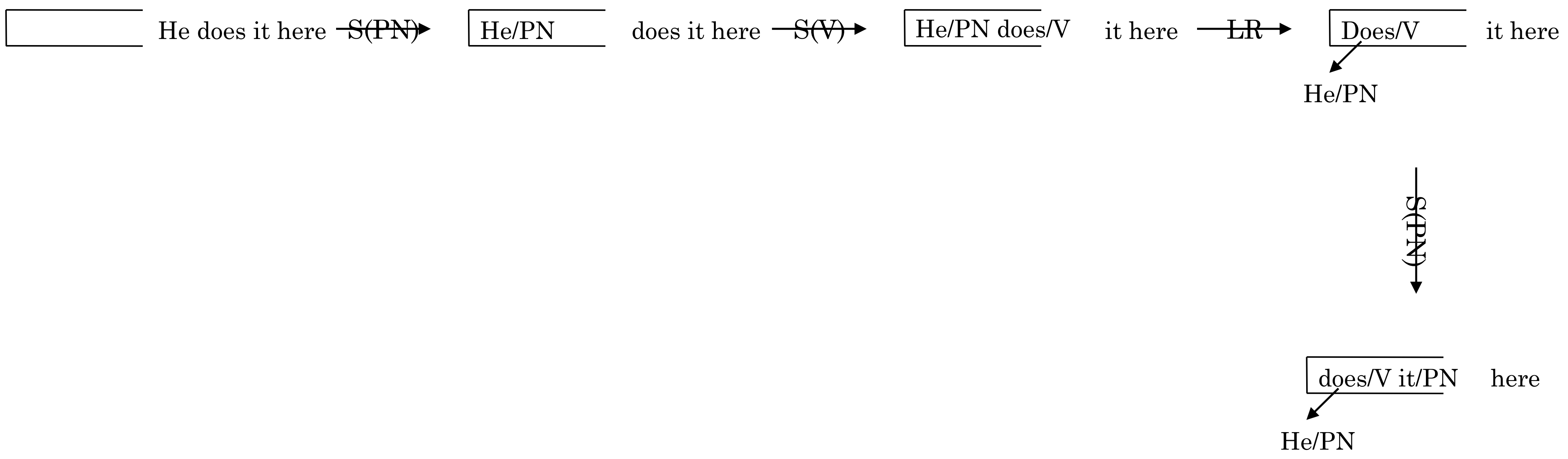
- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE



Hatori et al. IJCNLP 2011

The extended arc-standard transition system

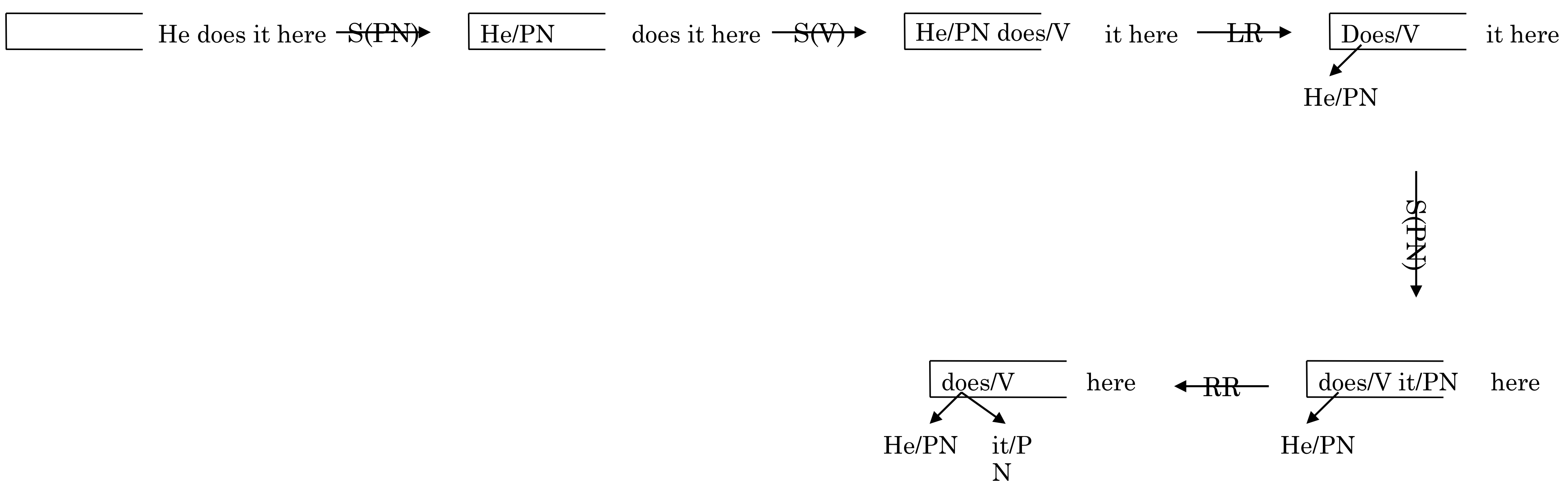
- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE



Hatori et al. IJCNLP 2011

The extended arc-standard transition system

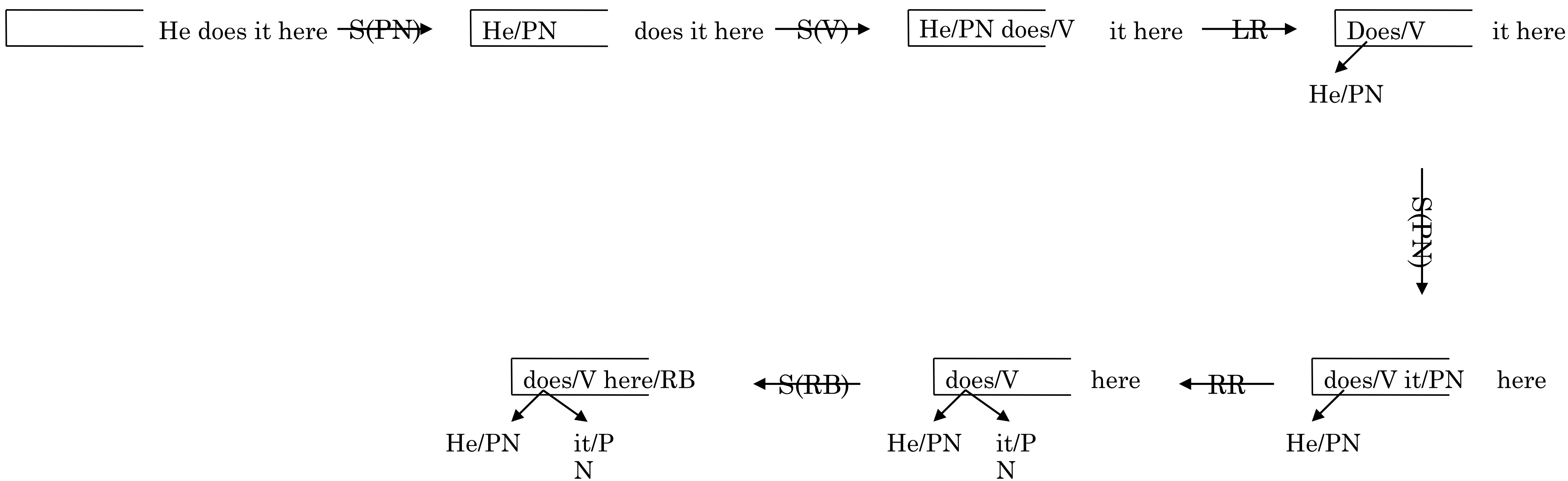
- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE



Hatori et al. IJCNLP 2011

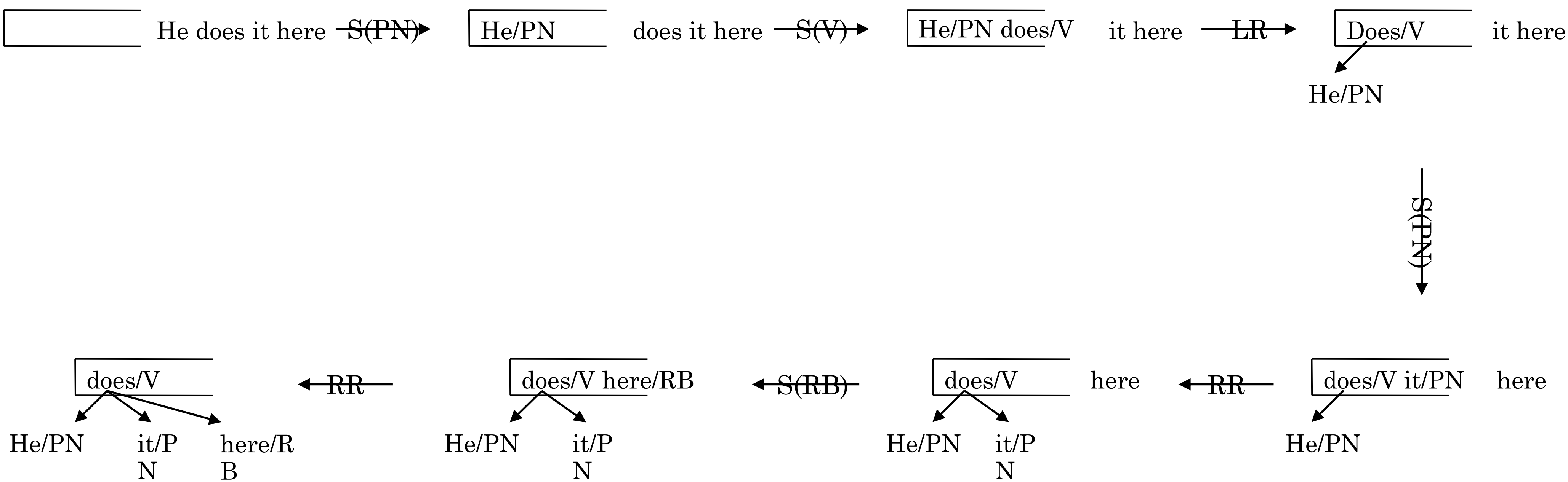
The extended arc-standard transition system

- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE



The extended arc-standard transition system

- An example
  - S(t) – SHIFT(t)
  - LR – LEFT-REDUCE
  - RR – RIGHT-REDUCE



Hatori et al. IJCNLP 2011

Features

POS tag features

$t \circ w_j$	$t \circ t_{j-1}$
$t \circ t_{j-1} \circ t_{j-2}$	$t \circ w_{j+1}$
$t \circ w_j \circ E(w_{j-1})$	$t \circ w_j \circ B(w_{j+1})$
$t \circ E(w_{j-1}) \circ w_j \circ B(w_{j+1})$ *(if $len(w_j) = 1$)*	
$t \circ B(w_j)$	$t \circ E(w_j)$
$t \circ C_n(w_j)$ $(n \in \{2, \ldots, len(w_j) - 1\})$	
$t \circ B(w_j) \circ C_n(w_j)$ $(n \in \{2, \ldots, len(w_j)\})$	
$t \circ E(w_j) \circ C_n(w_j)$ $(n \in \{1, \ldots, len(w_j) - 1\})$	
$t \circ C_n(w_j)$ *(if $C_n(w_j)$ equals to $C_{n+1}(w_j)$)*	
$t \otimes P(B(w_j))$	$t \otimes P(E(w_j))$

Hatori et al. IJCNLP 2011

Features

Dependency parsing features

(a)	$s_0.w$	$s_0.t$	$s_0.w \circ s_0.t$
	$s_1.w$	$s_1.t$	$s_1.w \circ s_1.t$
	$q_0.w$	$q_0.t$	$q_0.w \circ q_0.t$
	$s_0.w \circ s_1.w$		$s_0.t \circ s_1.t$
	$s_0.t \circ q_0.t$		$s_0.w \circ s_0.t \circ s_1.t$
	$s_0.t \circ s_1.w \circ s_1.t$		$s_0.w \circ s_1.w \circ s_1.t$
	$s_0.w \circ s_0.t \circ s_1.w$		$s_0.w \circ s_0.t \circ s_1.w \circ s_1.t$
	$s_0.t \circ q_0.t \circ q_1.t$		$s_1.t \circ s_0.t \circ q_0.t$
	$s_0.w \circ q_0.t \circ q_1.t$		$s_1.t \circ s_0.w \circ q_0.t$
	$s_1.t \circ s_1.\mathrm{rc}.t \circ s_0.t$		$s_1.t \circ s_1.\mathrm{lc}.t \circ s_0.t$
	$s_1.t \circ s_1.\mathrm{rc}.t \circ s_0.w$		$s_1.t \circ s_1.\mathrm{lc}.t \circ s_0.w$
	$s_1.t \circ s_0.t \circ s_0.\mathrm{rc}.t$		$s_1.t \circ s_0.w \circ s_0.\mathrm{lc}.t$
	$s_2.t \circ s_1.t \circ s_0.t$		

(b)	$s_0.w \circ d$	$s_0.t \circ d$	$s_1.w \circ d$	$s_1.w \circ d$
	$s_0.w \circ s_0.v_l$		$s_0.t \circ s_0.v_l$	
	$s_1.w \circ s_1.v_r$		$s_1.t \circ s_1.v_r$	
	$s_1.w \circ s_1.v_l$		$s_1.t \circ s_1.v_l$	
	$s_0.\mathrm{lc}.w$	$s_0.\mathrm{lc}.t$	$s_1.\mathrm{rc}.w$	$s_1.\mathrm{rc}.t$
	$s_1.\mathrm{lc}.w$	$s_1.\mathrm{lc}.t$	$s_0.\mathrm{lc}_2.w$	$s_0.\mathrm{lc}_2.t$
	$s_1.\mathrm{rc}_2.w$	$s_1.\mathrm{rc}_2.t$	$s_1.\mathrm{lc}_2.w$	$s_1.\mathrm{lc}_2.t$
	$s_0.t \circ s_0.\mathrm{lc}.t \circ s_0.\mathrm{lc}_2.t$		$s_1.t \circ s_1.\mathrm{rc}.t \circ s_1.\mathrm{rc}_2.t$	
	$s_1.t \circ s_1.\mathrm{lc}.t \circ s_1.\mathrm{lc}_2.t$			

Hatori et al. IJCNLP 2011

Features

Syntactic features

$t \circ s_0.w$	$t \circ s_0.t$
$t \circ s_0.w \circ q_0.w$	$t \circ s_0.t \circ q_0.w$
$t \circ B(s_0.w) \circ q_0.w$	$t \circ E(s_0.w) \circ q_0.w$
$t \circ s_0.t \circ s_0.\mathrm{rc}.t$	$t \circ s_0.t \circ s_0.\mathrm{lc}.t$
$t \circ s_0.w \circ s_0.t \circ s_0.\mathrm{rc}.t$	$t \circ s_0.w \circ s_0.t \circ s_0.\mathrm{lc}.t$

Hatori et al. IJCNLP 2011

Experiments

- CTB5 dataset

Training, development, and test data for Chinese dependency parsing.

	Sections	Sentences	Words
Training	001–815 1,001–1,136	16,118	437,859
Dev	886–931 1,148–1,151	804	20,453
Test	816–885 1,137–1,147	1,915	50,319

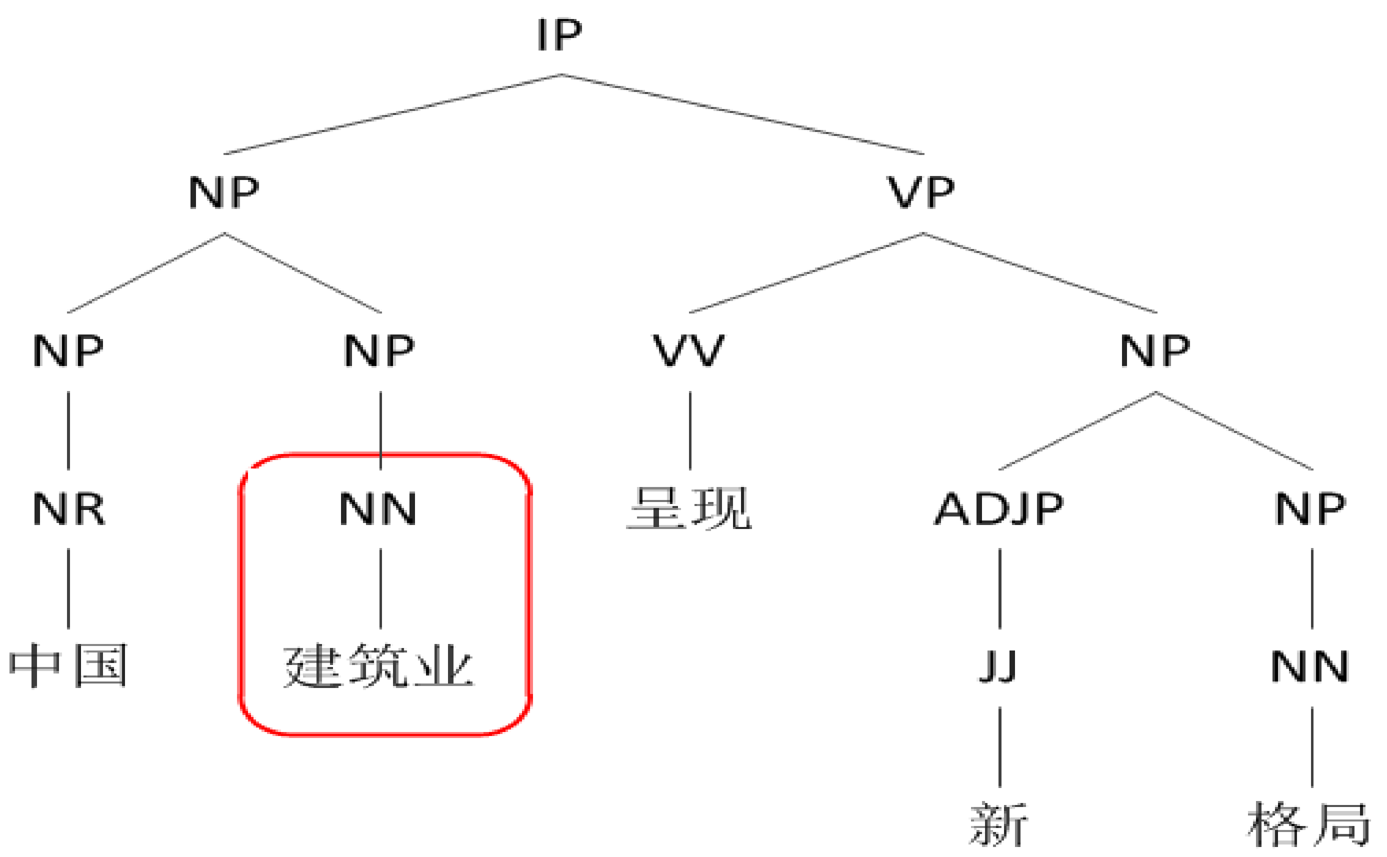
Results

Model	LAS	UAS	POS
Li et al. (2011) (unlabeled)		80.74	93.08
Li et al. (2012) (unlabeled)	---	81.21	94.51
Li et al. (2012) (labeled)	79.01	81.67	94.60
Hatori et al. (2011) (unlabeled)	---	81.33	93.94
Bohnet and Nirve (2012) (labeled)	77.91	81.42	93.24
Our implementation (unlabeled)	**---**	**81.20**	**94.15**
Out implementation (labeled)	**78.30**	**81.26**	**94.28**

Applications

- Word segmentation
- Dependency parsing
- Context free grammar parsing
- Combinatory categorial grammar parsing
- Joint segmentation and POS-tagging
- Joint POS-tagging and dependency parsing
- Joint segmentation, POS-tagging and constituent parsing
- Joint segmentation, POS-tagging and dependency parsing

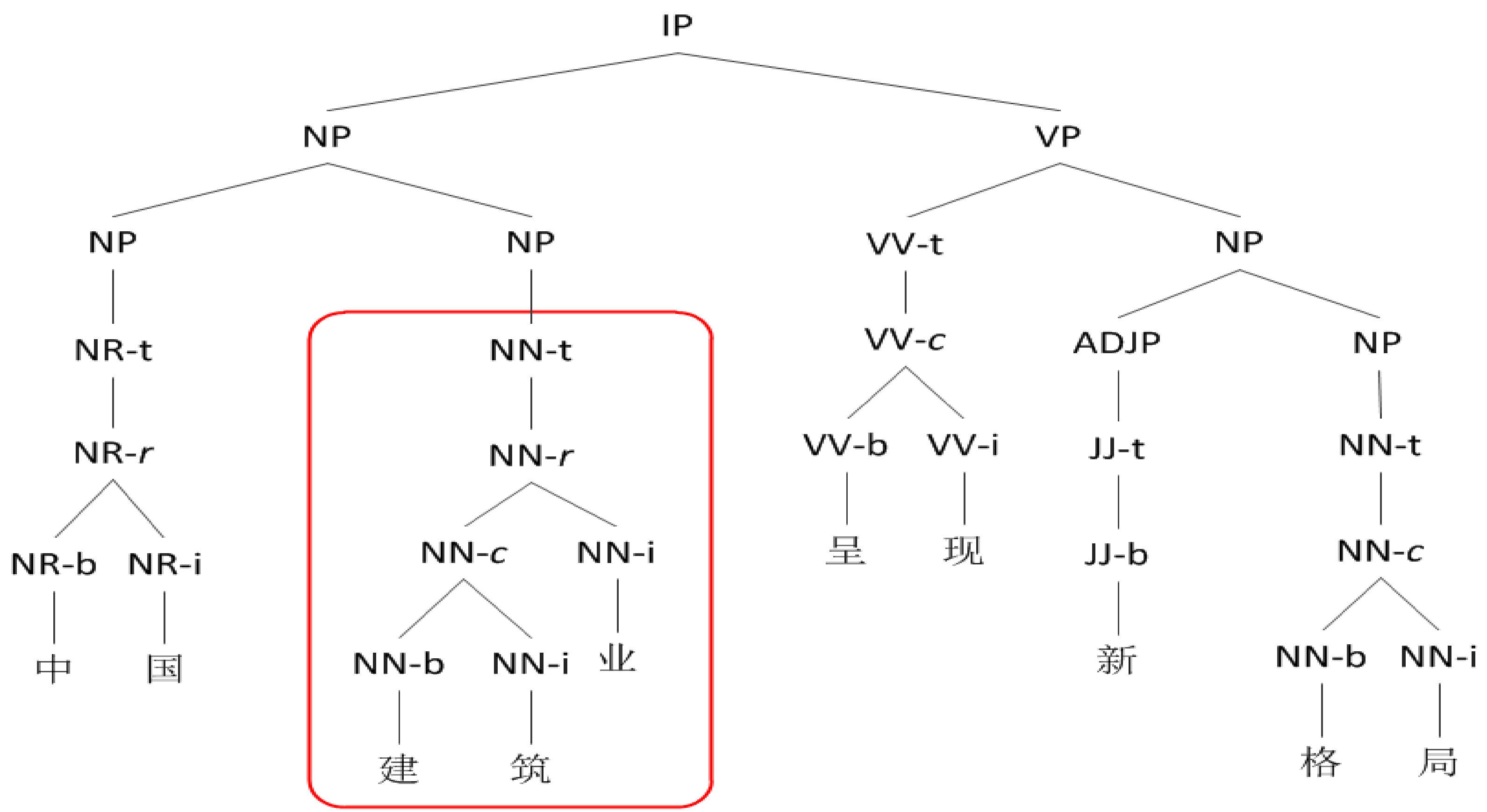
Traditional: word-based Chinese parsing



CTB-style word-based syntax tree for “中国 (China) 建筑业 (architecture industry) 呈现 (show) 新 (new) 格局 (pattern)”.

Zhang et al. ACL 2013

This: character-based Chinese parsing

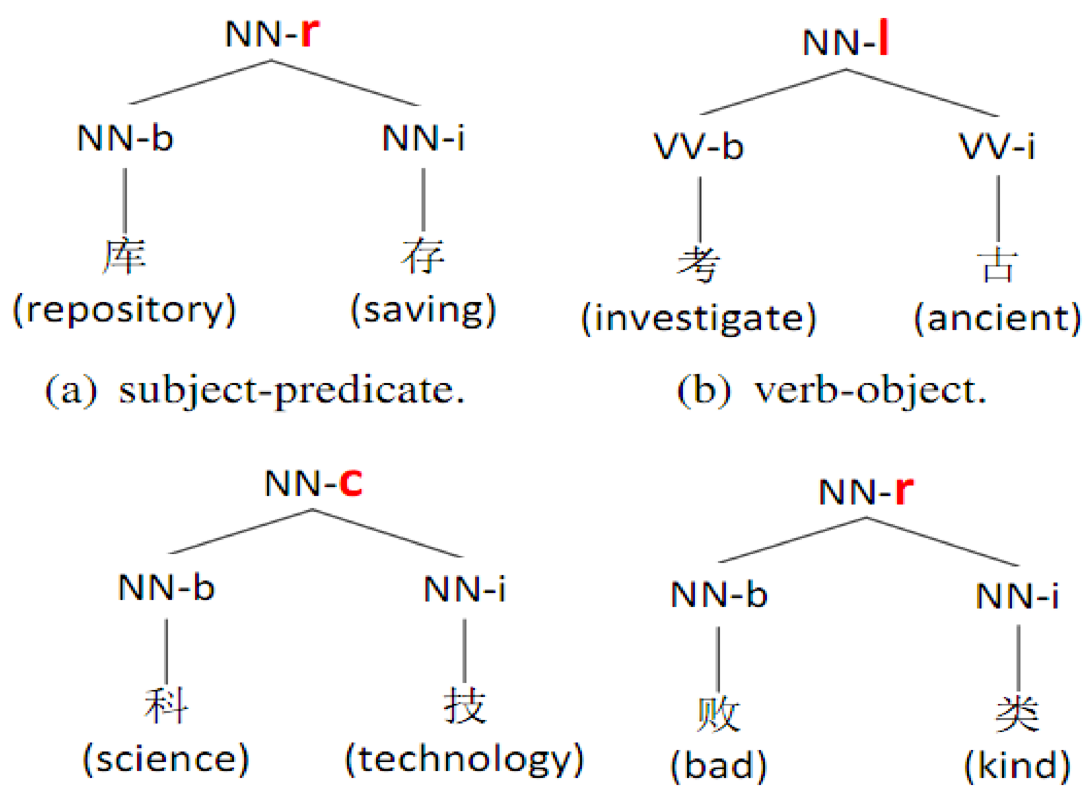


Character-level syntax tree with hierarchal word structures for “中 (middle) 国 (nation) 建 (construction) 筑 (building) 业 (industry) 呈 (present) 现 (show) 新 (new) 格 (style) 局 (situation)”.

Zhang et al. ACL 2013

Why character-based?

- Chinese words have syntactic structures.



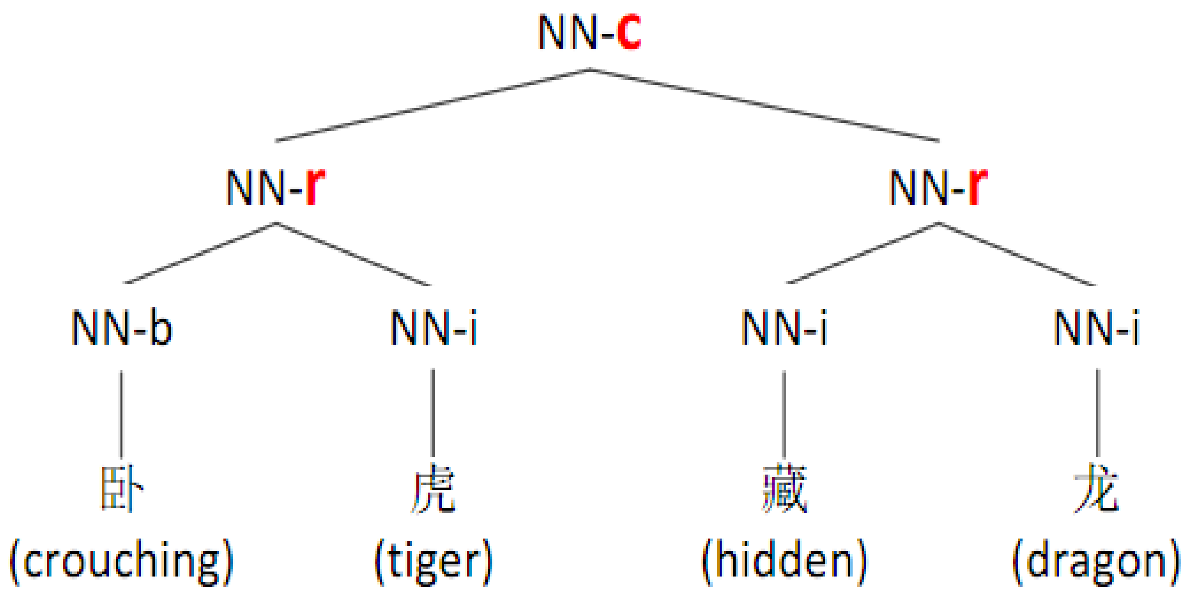
(a) subject-predicate. (b) verb-object.

(c) coordination. (d) modifier-noun.

Zhang et al. ACL 2013

Why character-based?

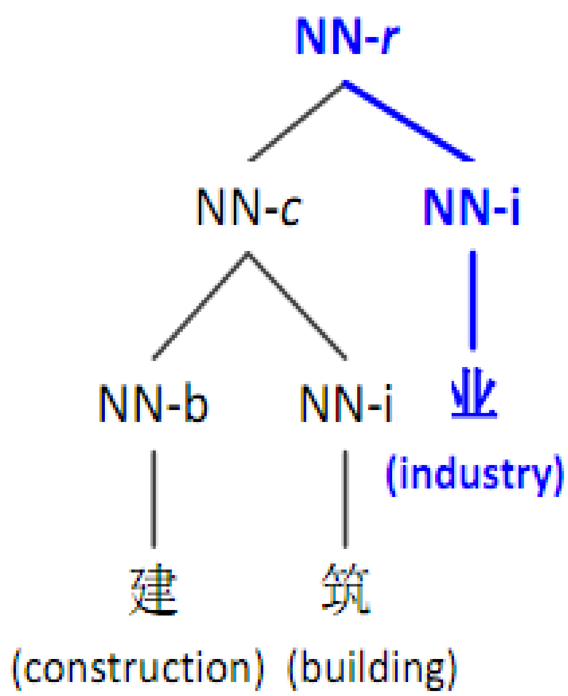
- Chinese words have syntactic structures.



Zhang et al. ACL 2013

Why character-based?

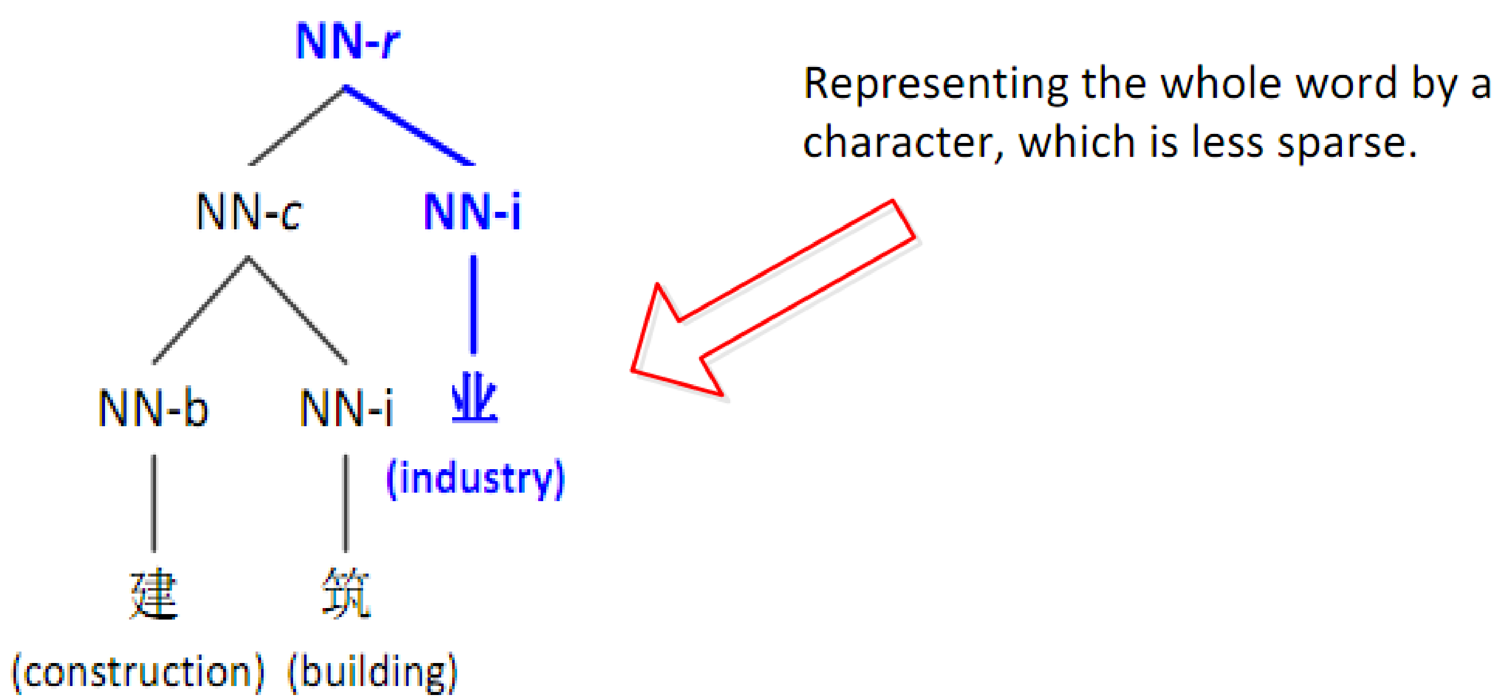
- Deep character information of word structures.



Zhang et al. ACL 2013

Why character-based?

- Deep character information of word structures.



Zhang et al. ACL 2013

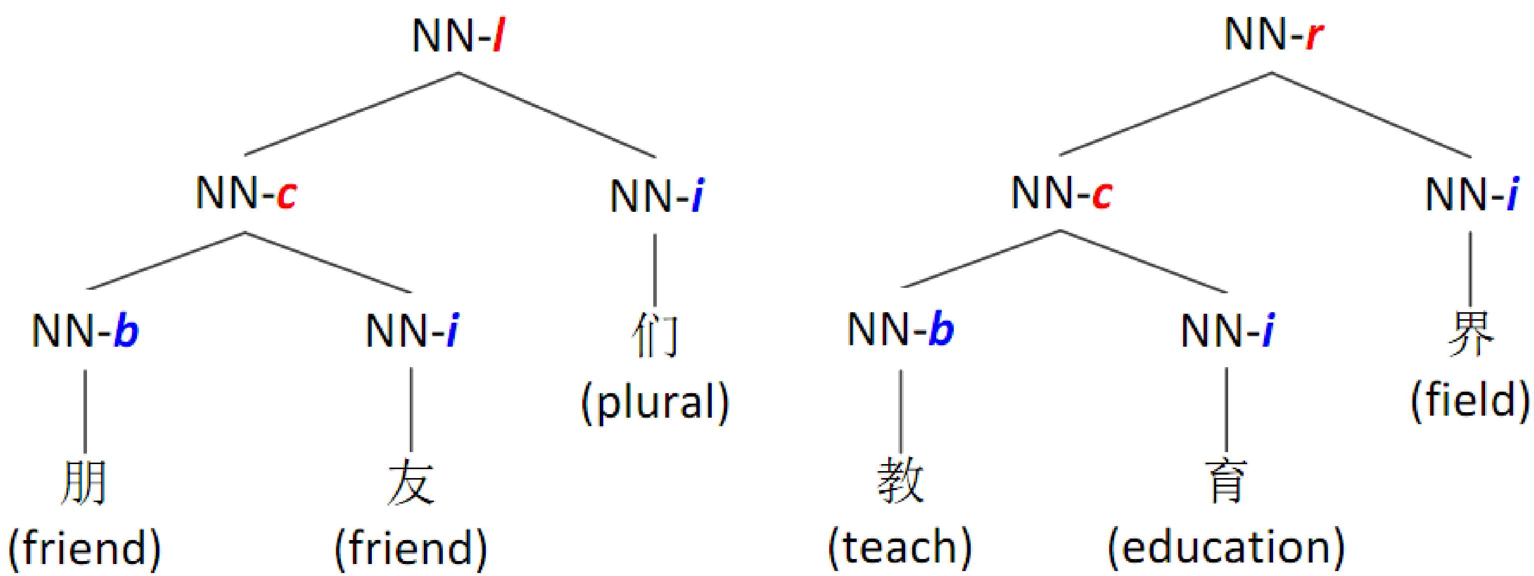
Why character-based?

- Build syntax tree from character sequences.
  - Not require segmentation or POS-tagging as input.
  - Benefit from joint framework, avoid error propagation.

Zhang et al. ACL 2013

Word structure annotation

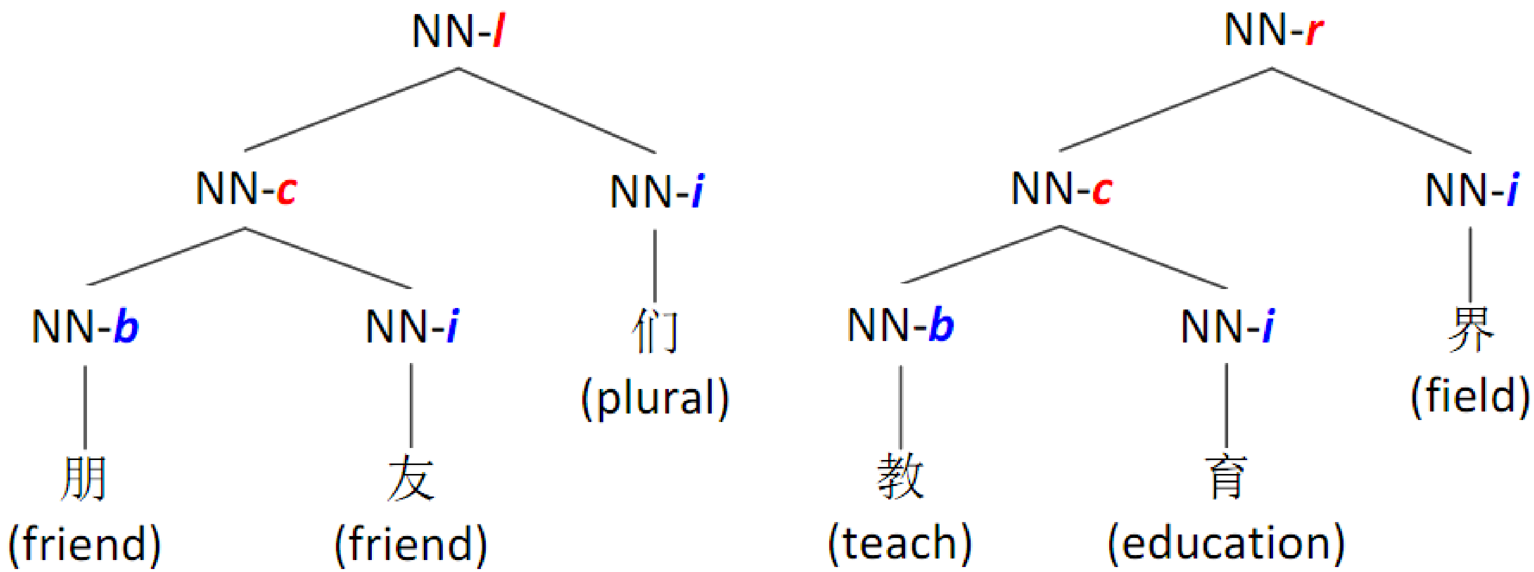
- Binarized tree structure for each word.



Zhang et al. ACL 2013

Word structure annotation

- Binarized tree structure for each word.

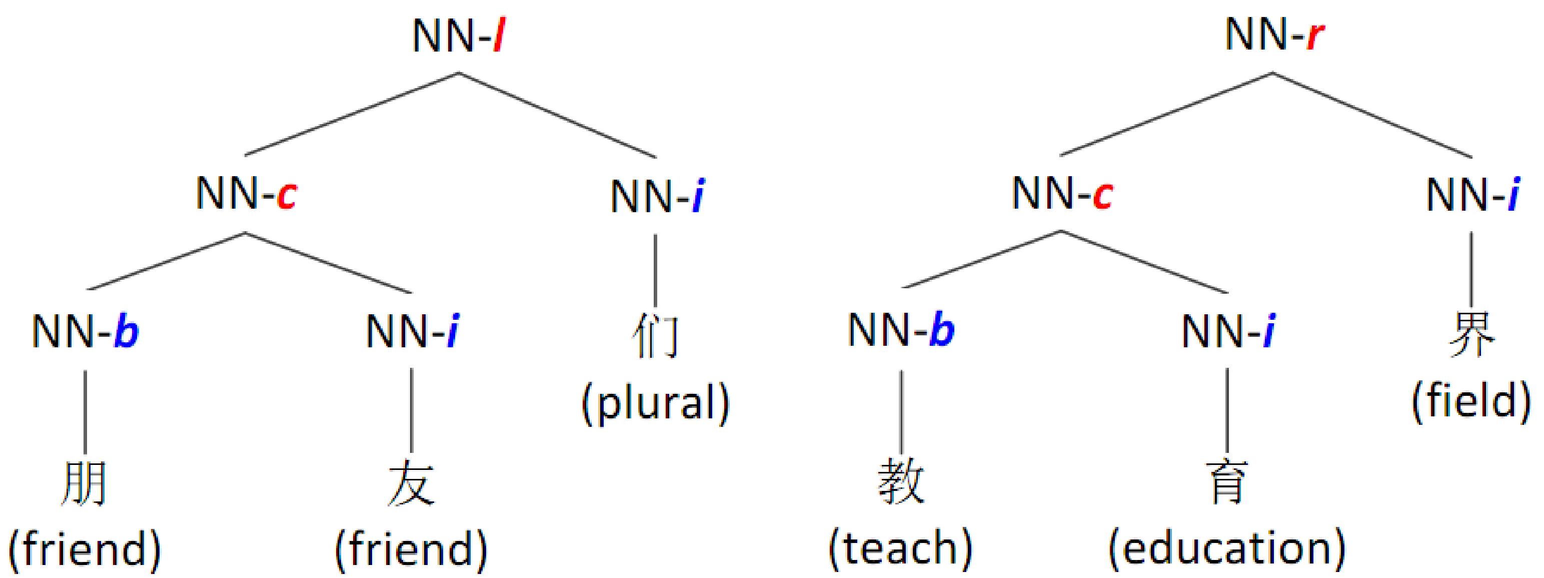


- **b, i denote whether the below character is at a word's beginning position.**
- **l, r, c denote the head direction of current node, respectively left, right and coordination.**

Zhang et al. ACL 2013

Word structure annotation

- Binarized tree structure for each word.



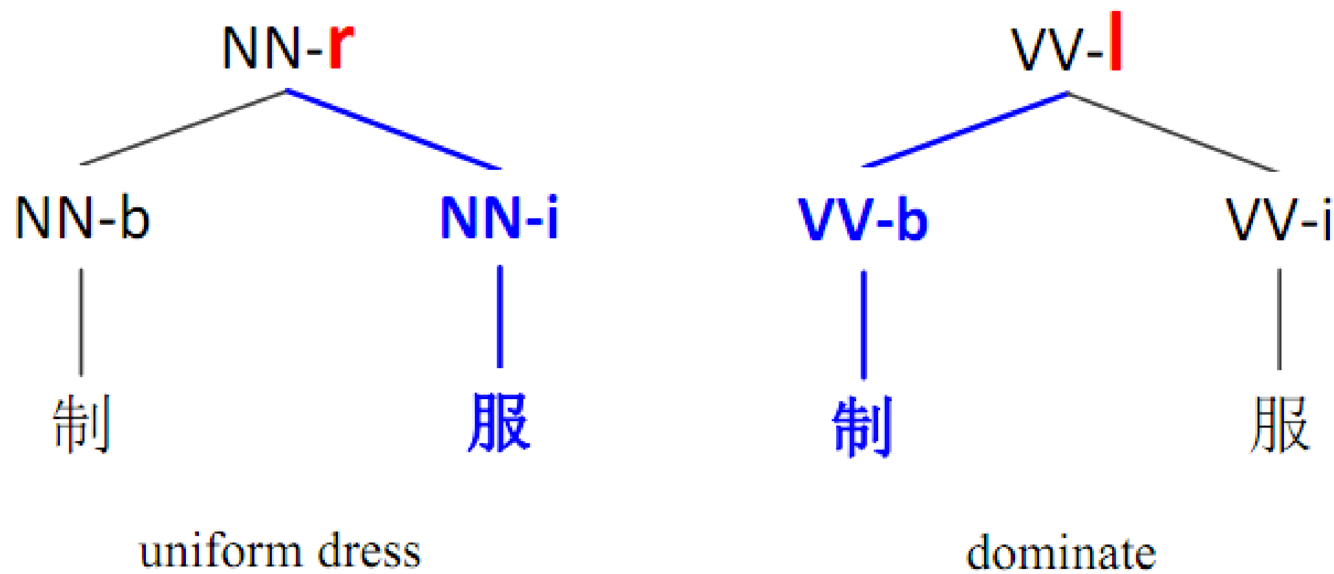
- **b, i denote whether the below character is at a word's beginning position.**
- **l, r, c denote the head direction of current node, respectively left, right and coordination.**

We extend word-based phrase-structures into character-based syntax trees using the word structures demonstrated above.

Zhang et al. ACL 2013

Word structure annotation

- Annotation input: a word and its POS.
  - A word may have different structures according to different POS.



Zhang et al. ACL 2013

The character-based parsing model

- A transition-based parser

Zhang et al. ACL 2013

The character-based parsing model

- A transition-based parser
  - Extended from Zhang and Clark (2009), a word-based transition parser.

Zhang et al. ACL 2013

The character-based parsing model

- A transition-based parser
  - Extended from Zhang and Clark (2009), a word-based transition parser.
- Incorporating features of a word-based parser as well as a joint SEG&POS system.

Zhang et al. ACL 2013

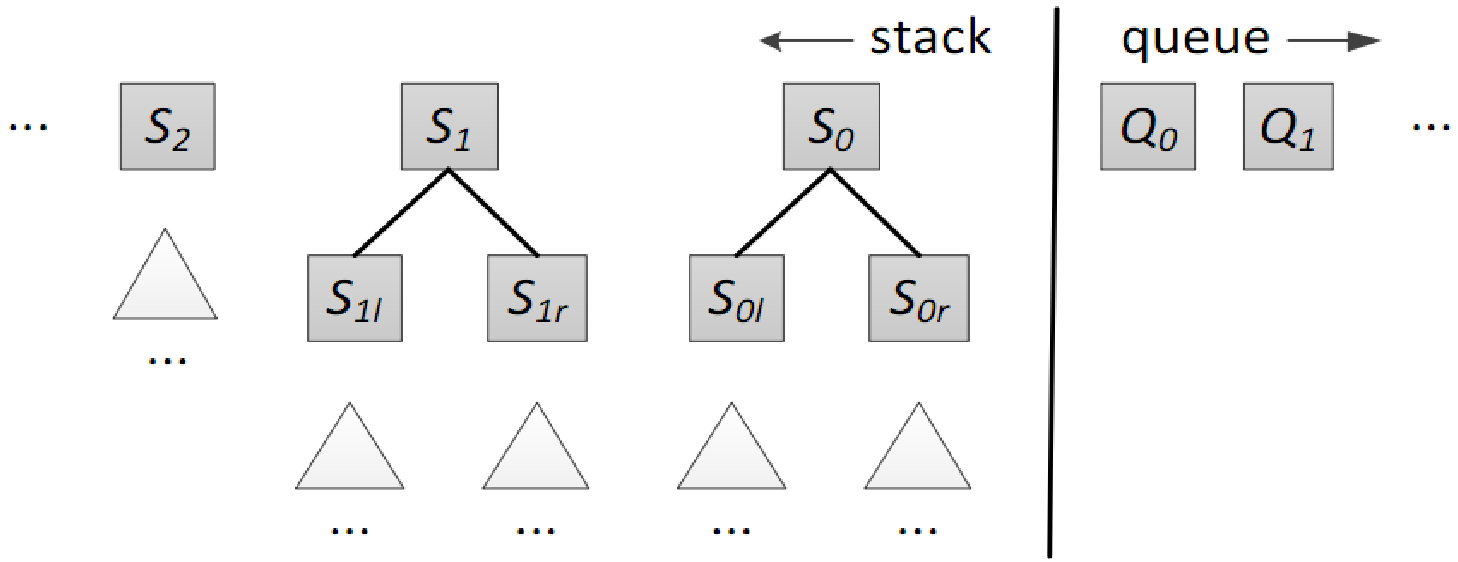
The character-based parsing model

- A transition-based parser
  - Extended from Zhang and Clark (2009), a word-based transition parser.
- Incorporating features of a word-based parser as well as a joint SEG&POS system.
- Adding the deep character information from word structures.

Zhang et al. ACL 2013

The transition system

- State:

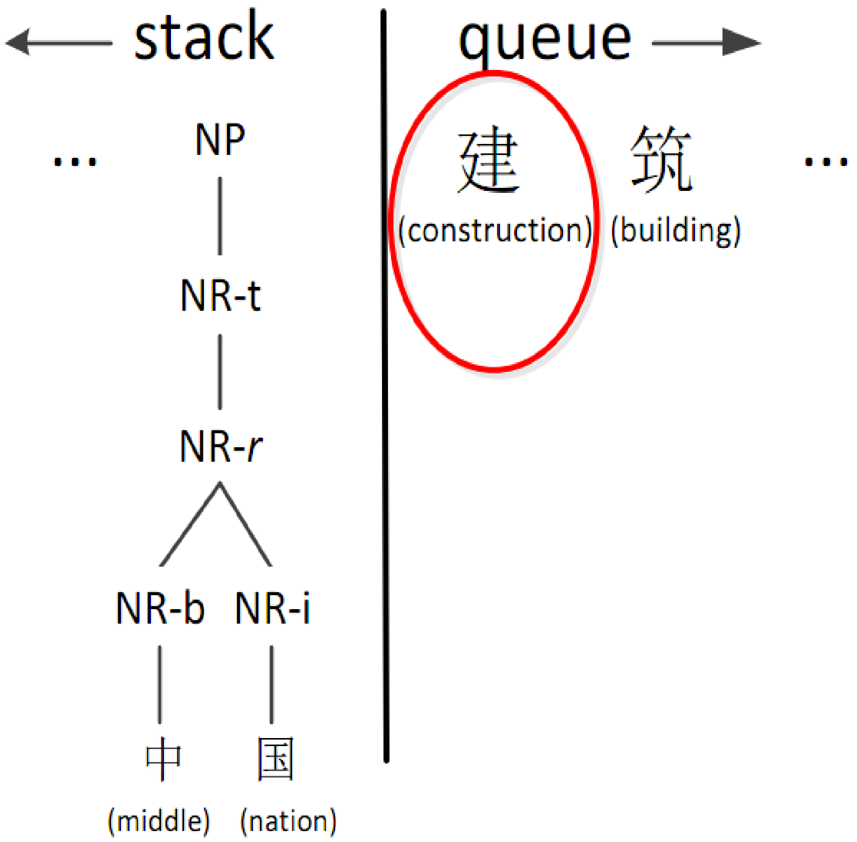


- Actions:
  - SHIFT-SEPARATE(***t***), SHIFT-APPEND, REDUCE-SUBWORD(***d***), REDUCE-WORD, REDUCE-BINARY(***d***;***l***), REDUCE-UNARY(***l***), TERMINATE

Zhang et al. ACL 2013

Actions

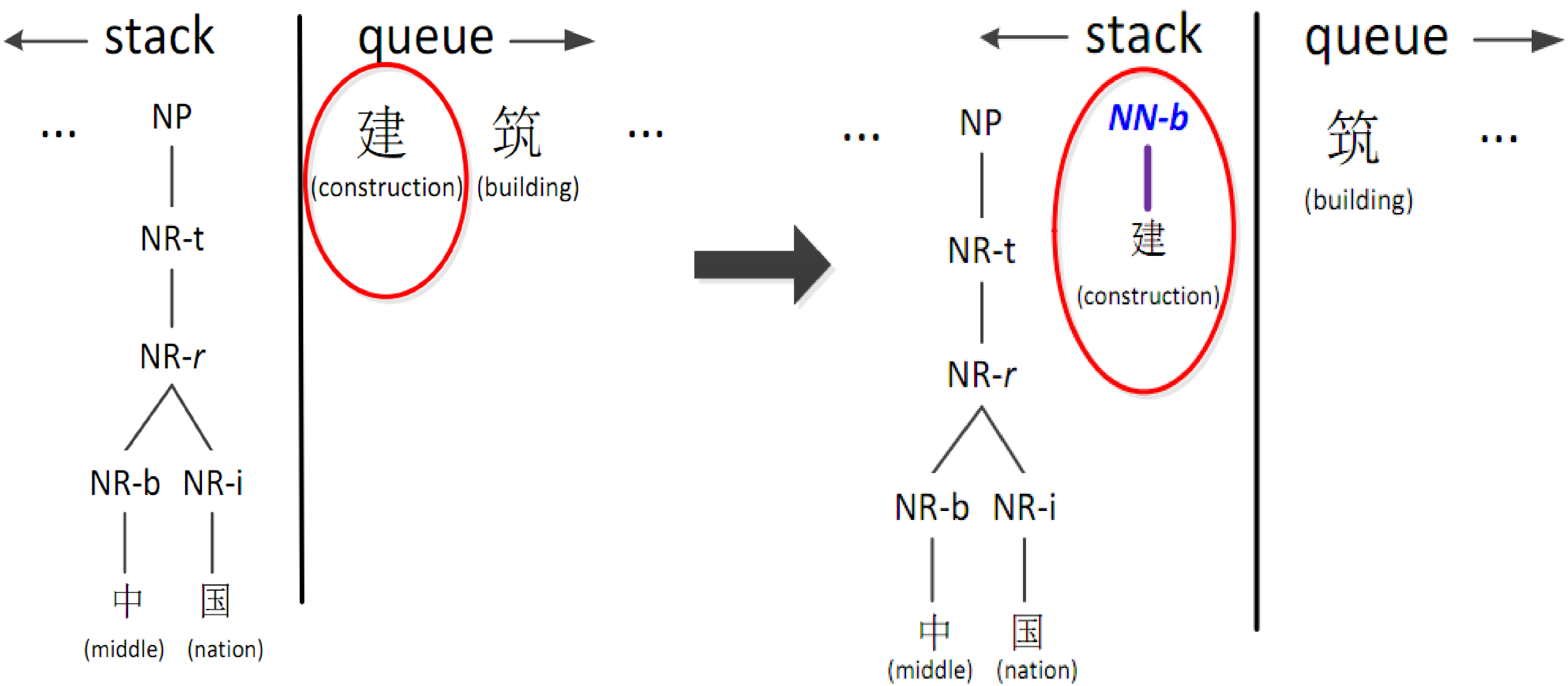
- SHIFT-SEPARATE(t)



Zhang et al. ACL 2013

Actions

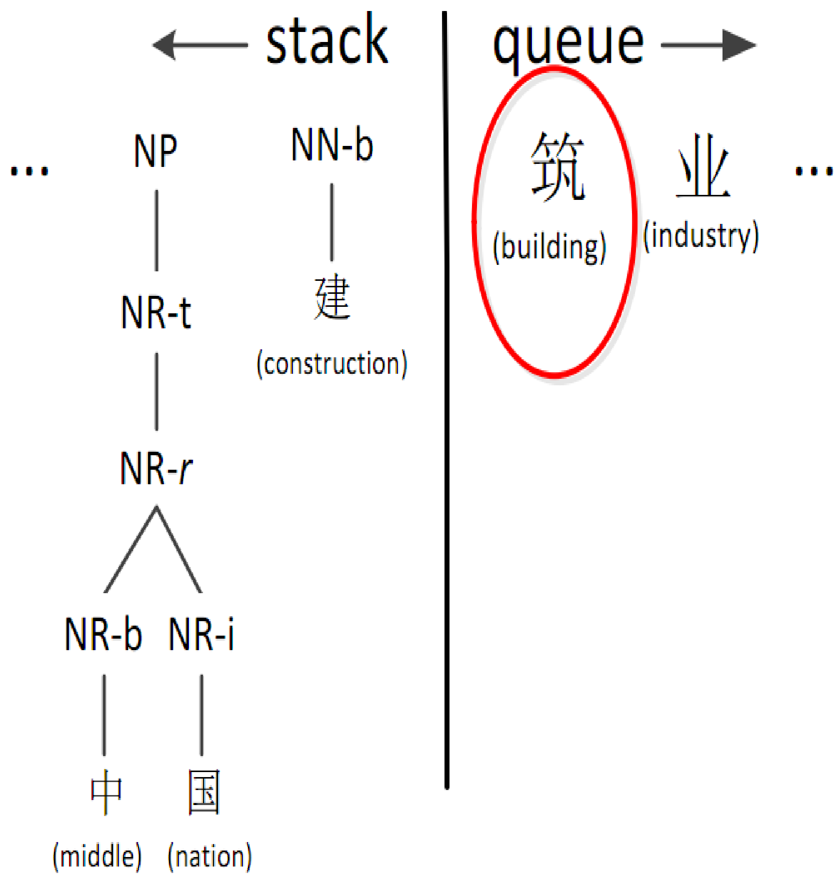
- SHIFT-SEPARATE(t)



Zhang et al. ACL 2013

Actions

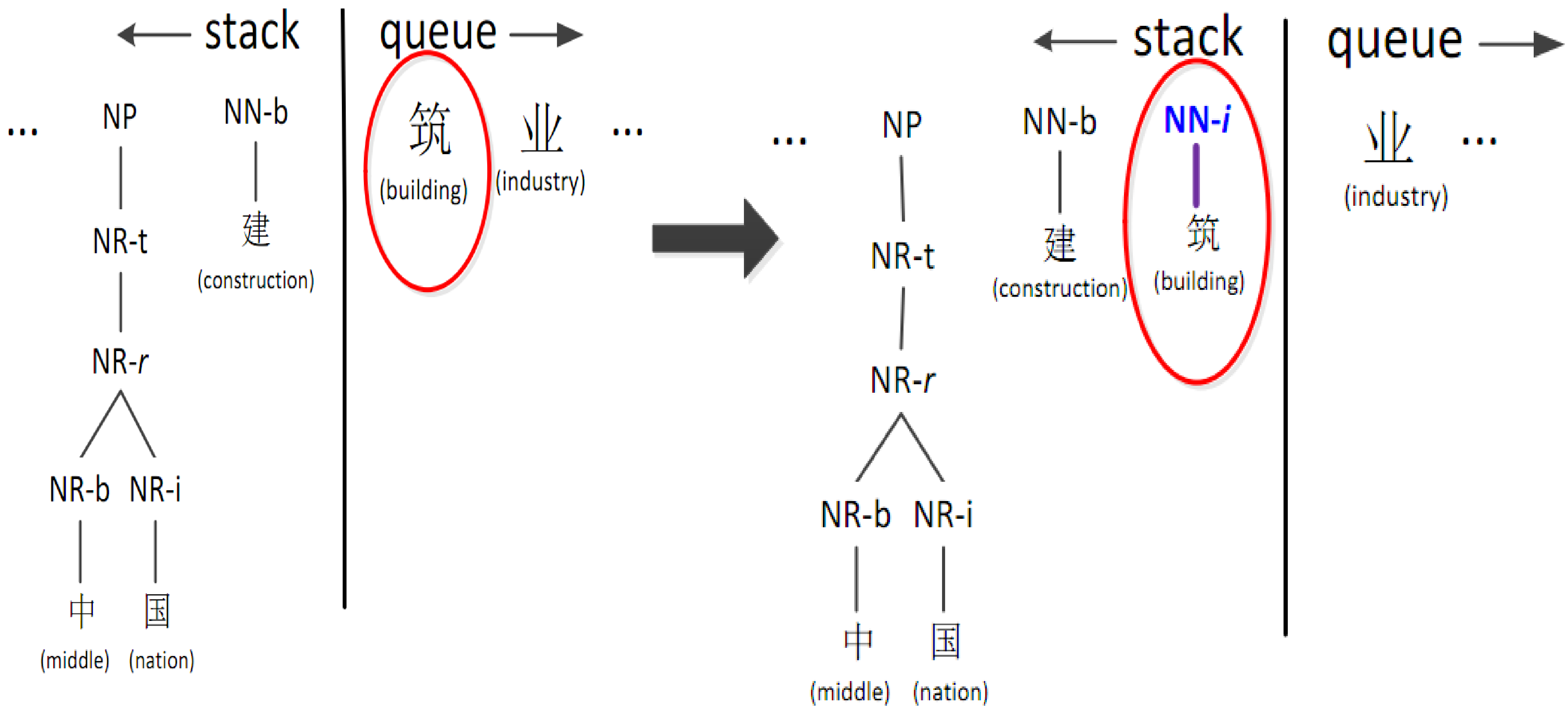
- SHIFT-APPEND



Zhang et al. ACL 2013

Actions

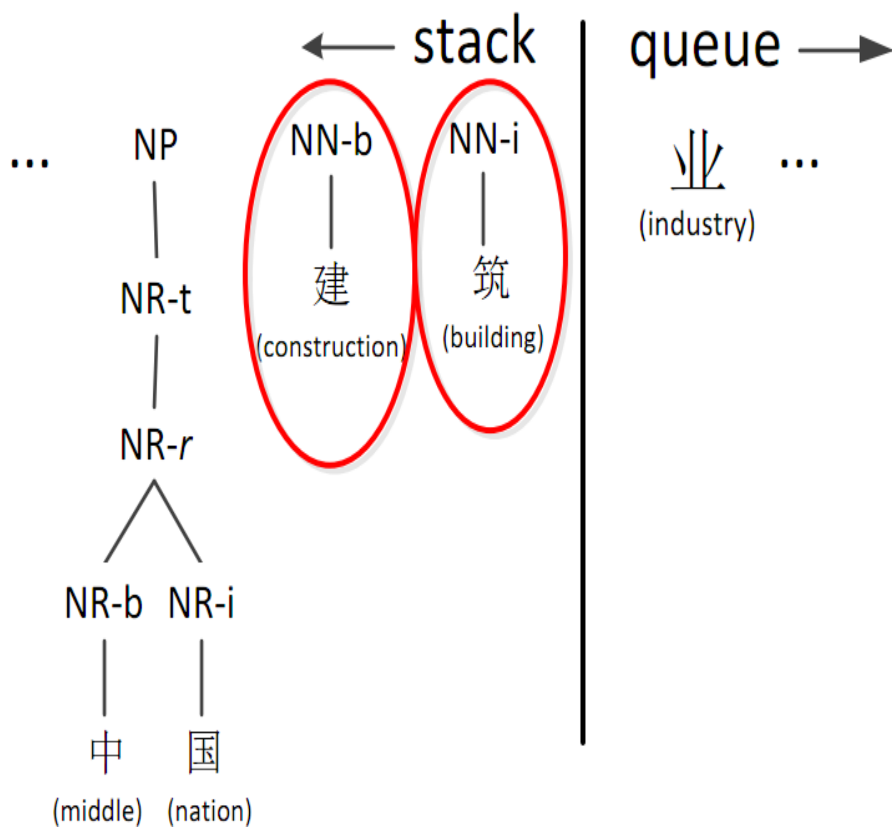
- SHIFT-APPEND



Zhang et al. ACL 2013

Actions

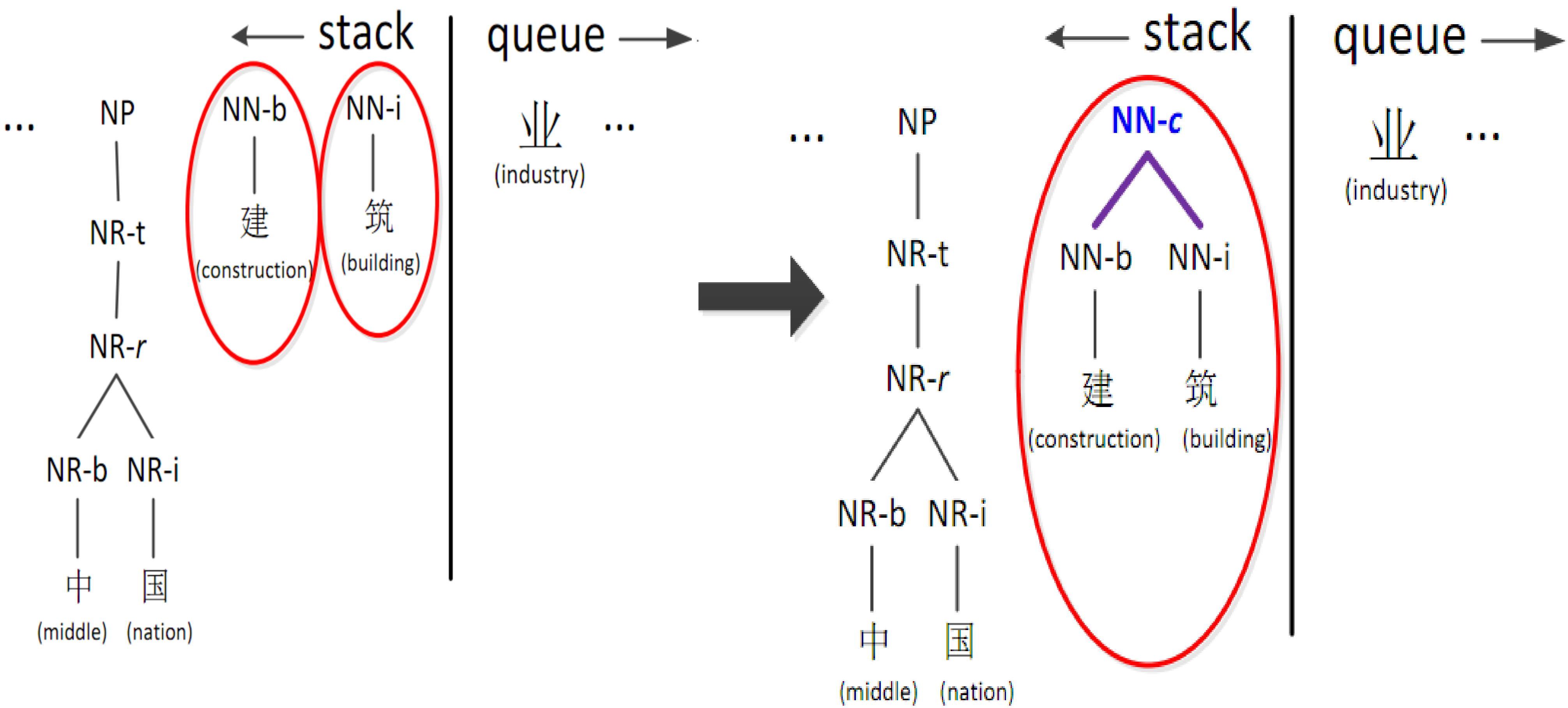
- REDUCE-SUBWORD(d)



Zhang et al. ACL 2013

Actions

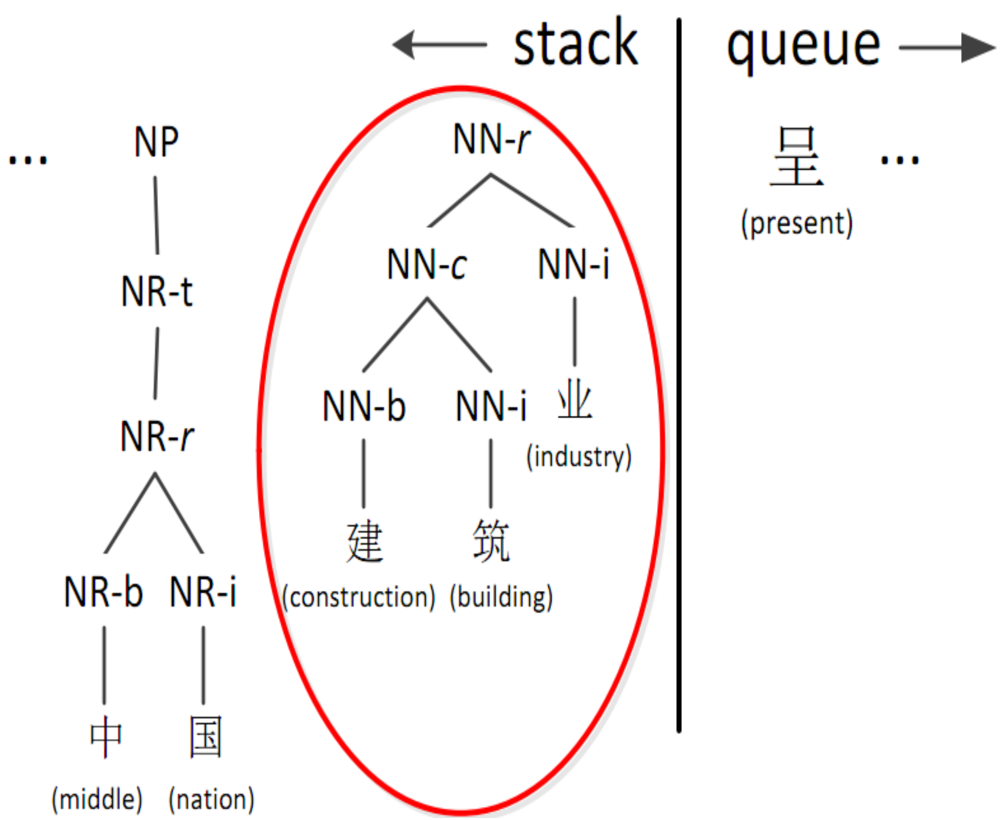
- REDUCE-SUBWORD(d)



Zhang et al. ACL 2013

Actions

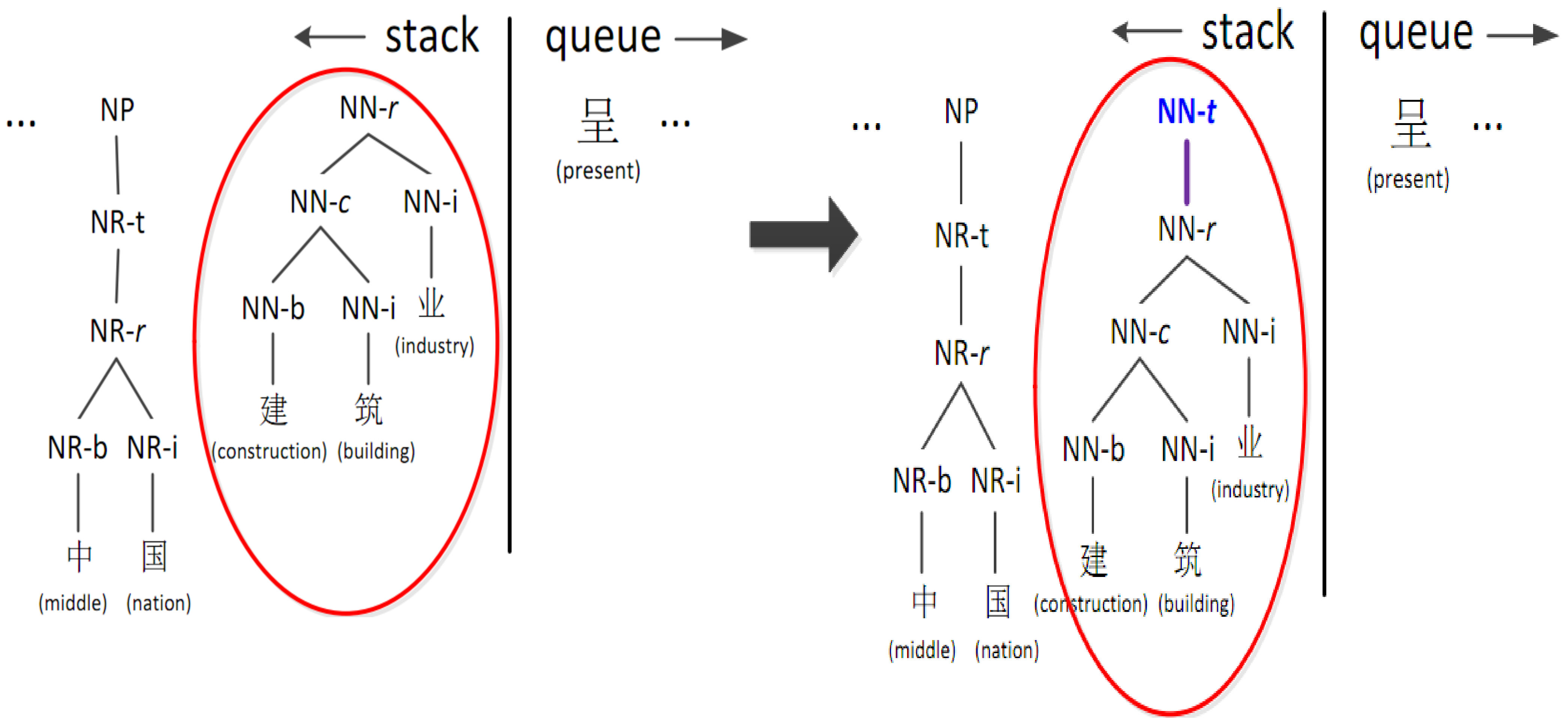
- REDUCE-WORD



Zhang et al. ACL 2013

Actions

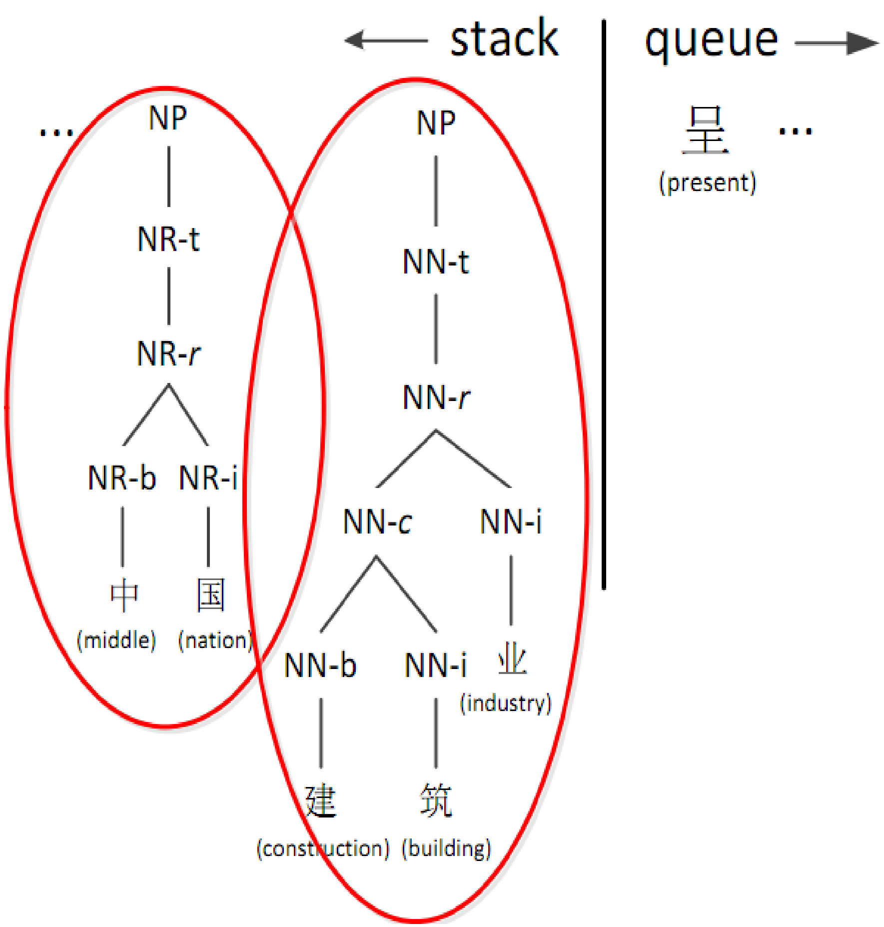
- REDUCE-WORD



Zhang et al. ACL 2013

Actions

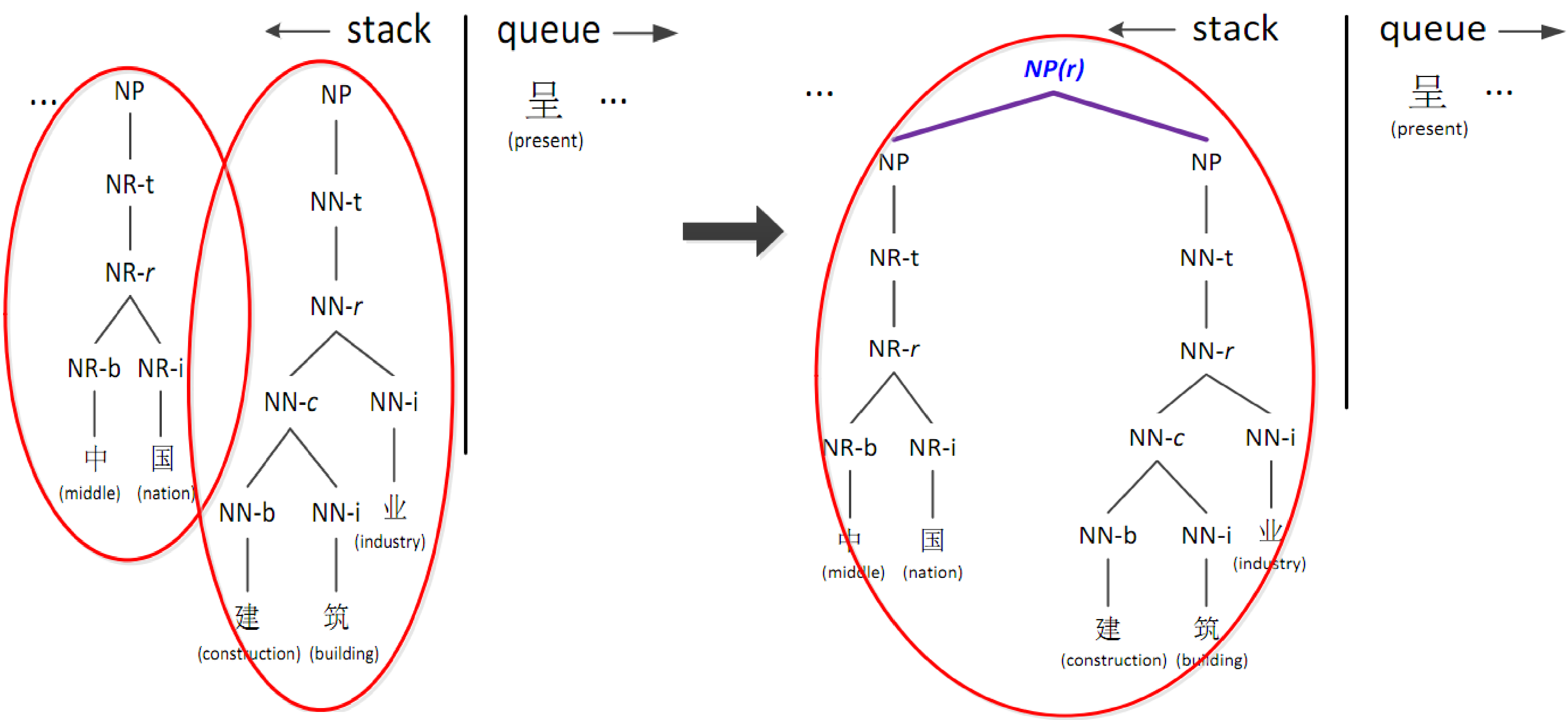
- REDUCE-BINARY(d; l)



Zhang et al. ACL 2013

Actions

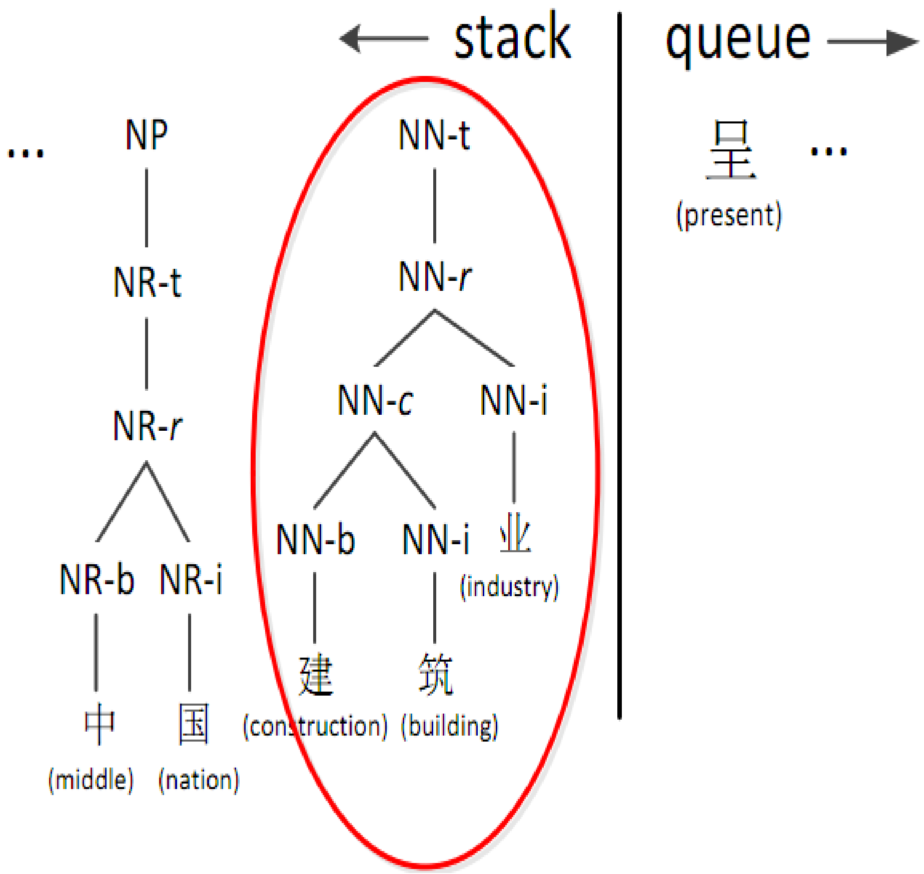
- REDUCE-BINARY(d; l)



Zhang et al. ACL 2013

Actions

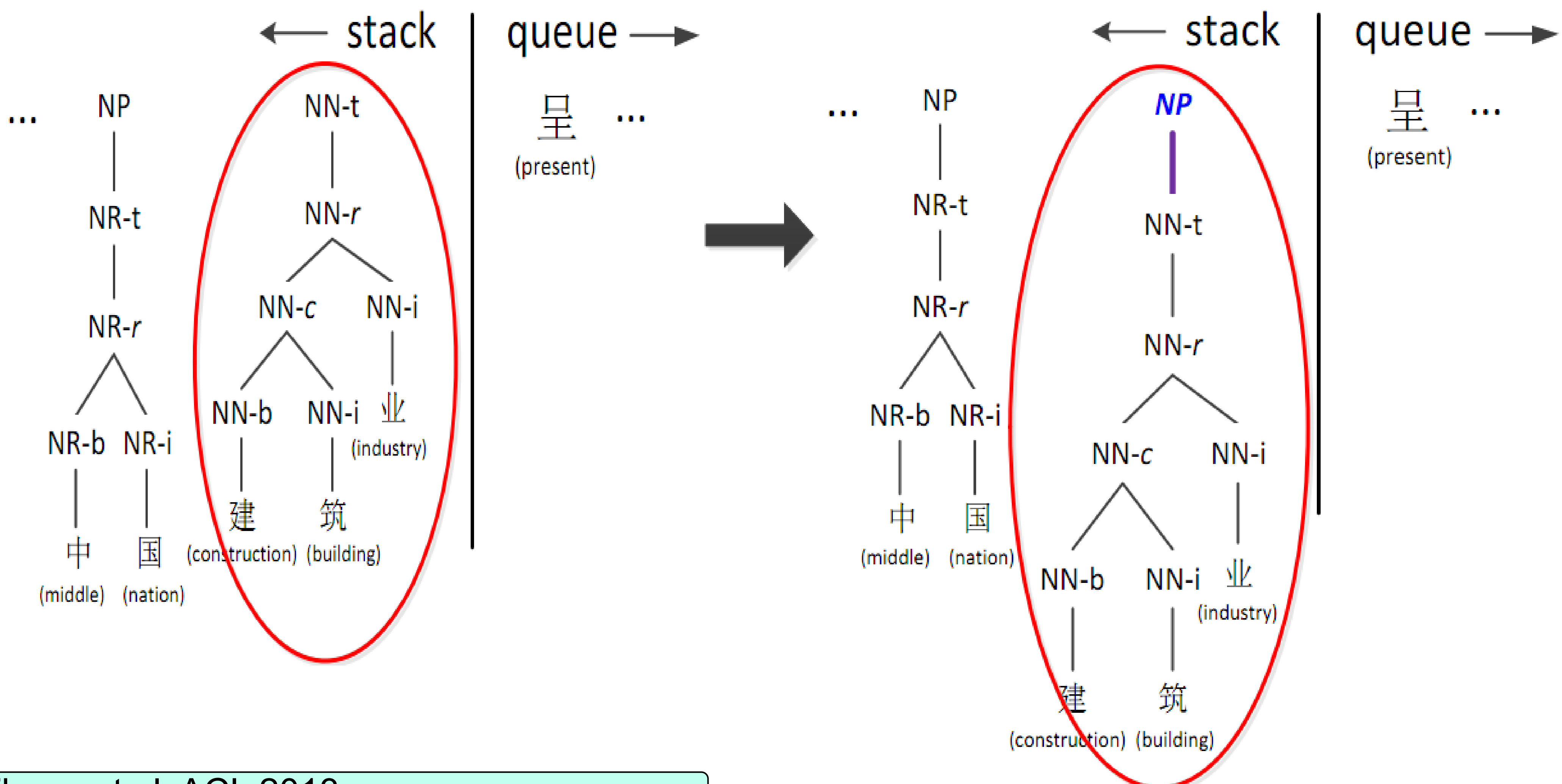
- REDUCE-UNARY(l)



Zhang et al. ACL 2013

Actions

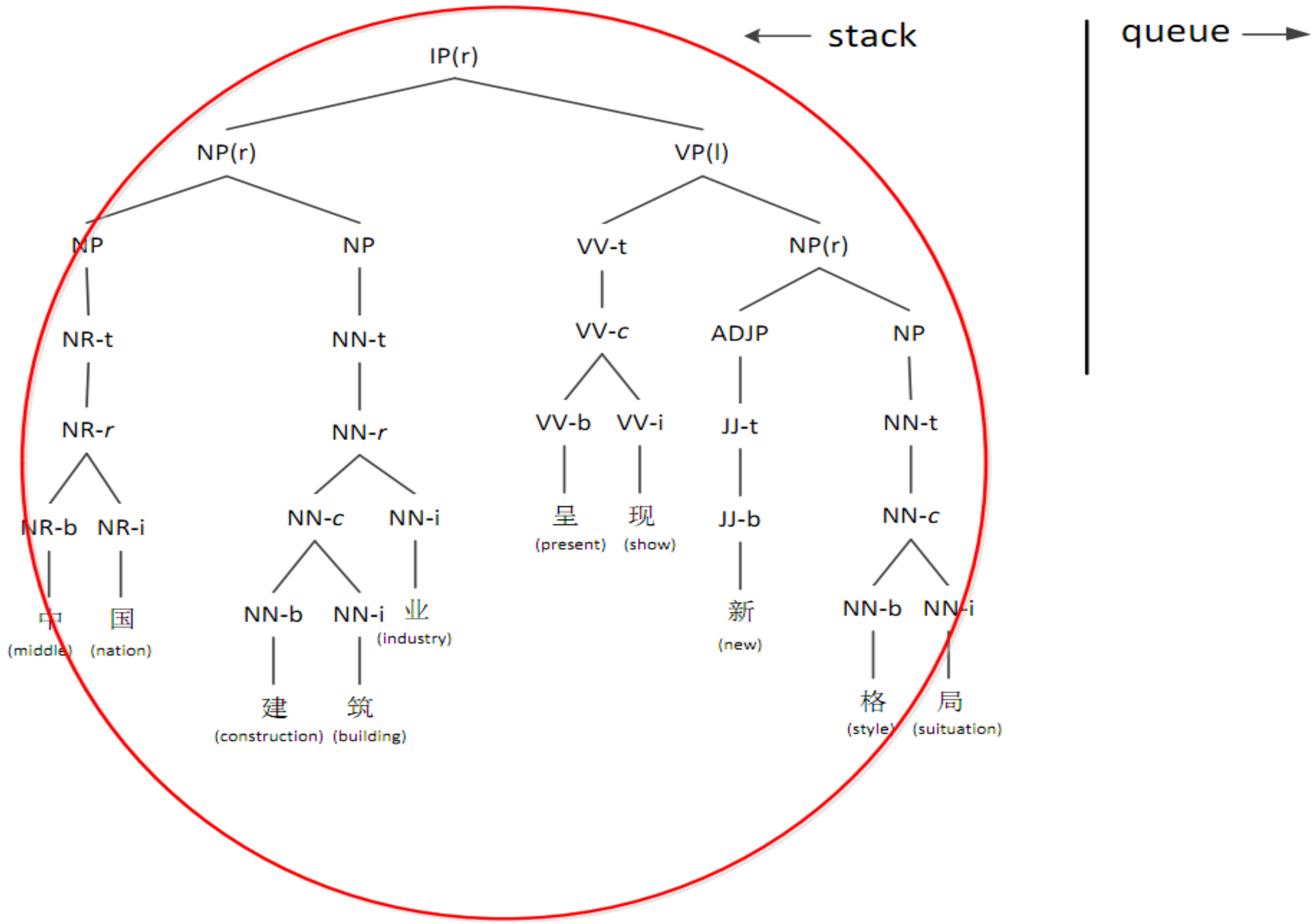
- REDUCE-UNARY(l)



Zhang et al. ACL 2013

Actions

- TERMINATE



Zhang et al. ACL 2013

Features

- From word-based parser (Zhang and Clark, 2009)
- From joint SEG&POS-Tagging (Zhang and Clark, 2010)

Features

- From word-based parser (Zhang and Clark, 2009)
- From joint SEG&POS-Tagging (Zhang and Clark, 2010)

baseline features

Features

- From word-based parser (Zhang and Clark, 2009)
- From joint SEG&POS-Tagging (Zhang and Clark, 2010)

baseline features

- Deep character features

Zhang et al. ACL 2013

Features

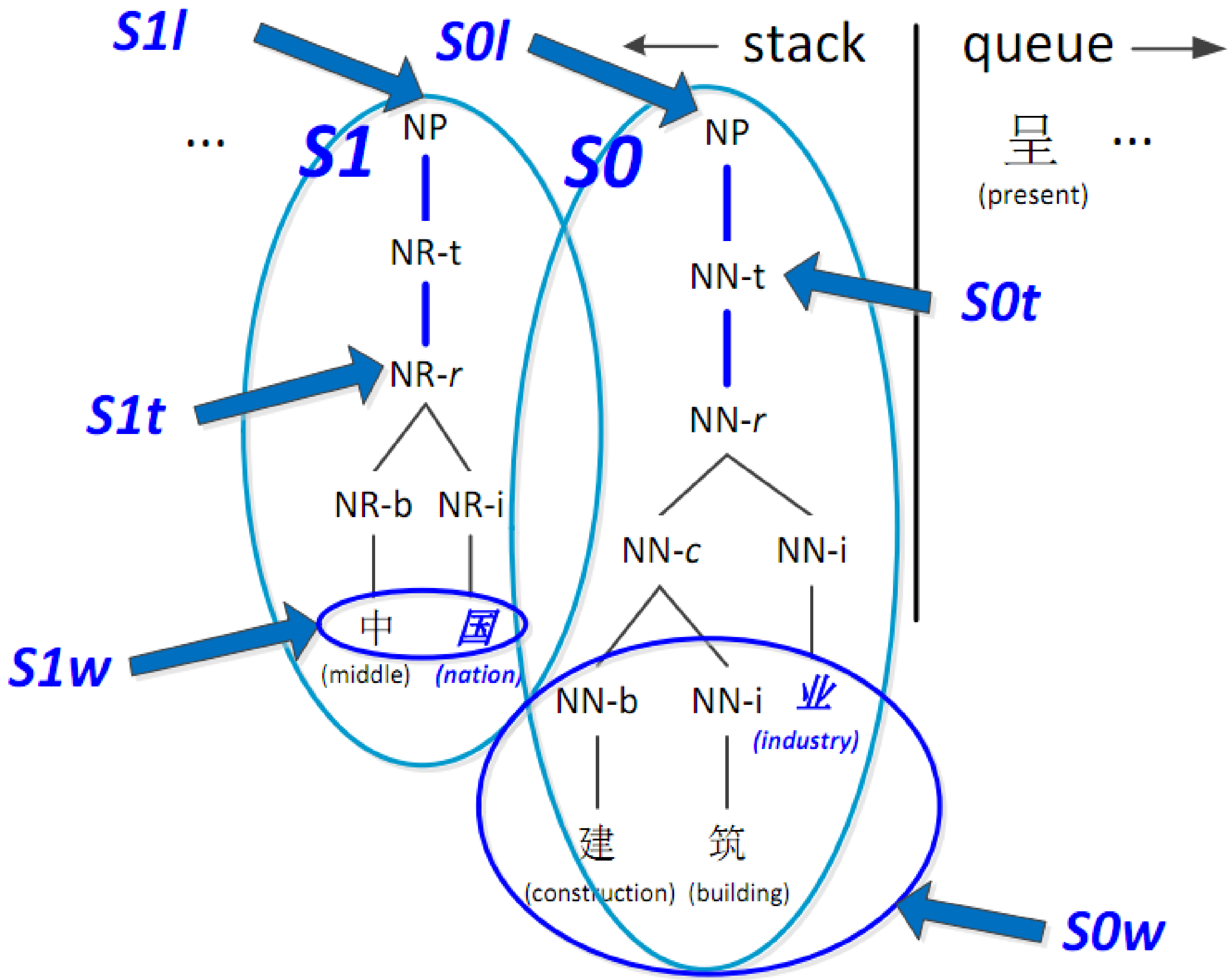
- From word-based parser (Zhang and Clark, 2009)
- From joint SEG&POS-Tagging (Zhang and Clark, 2010)

baseline features

- Deep character features

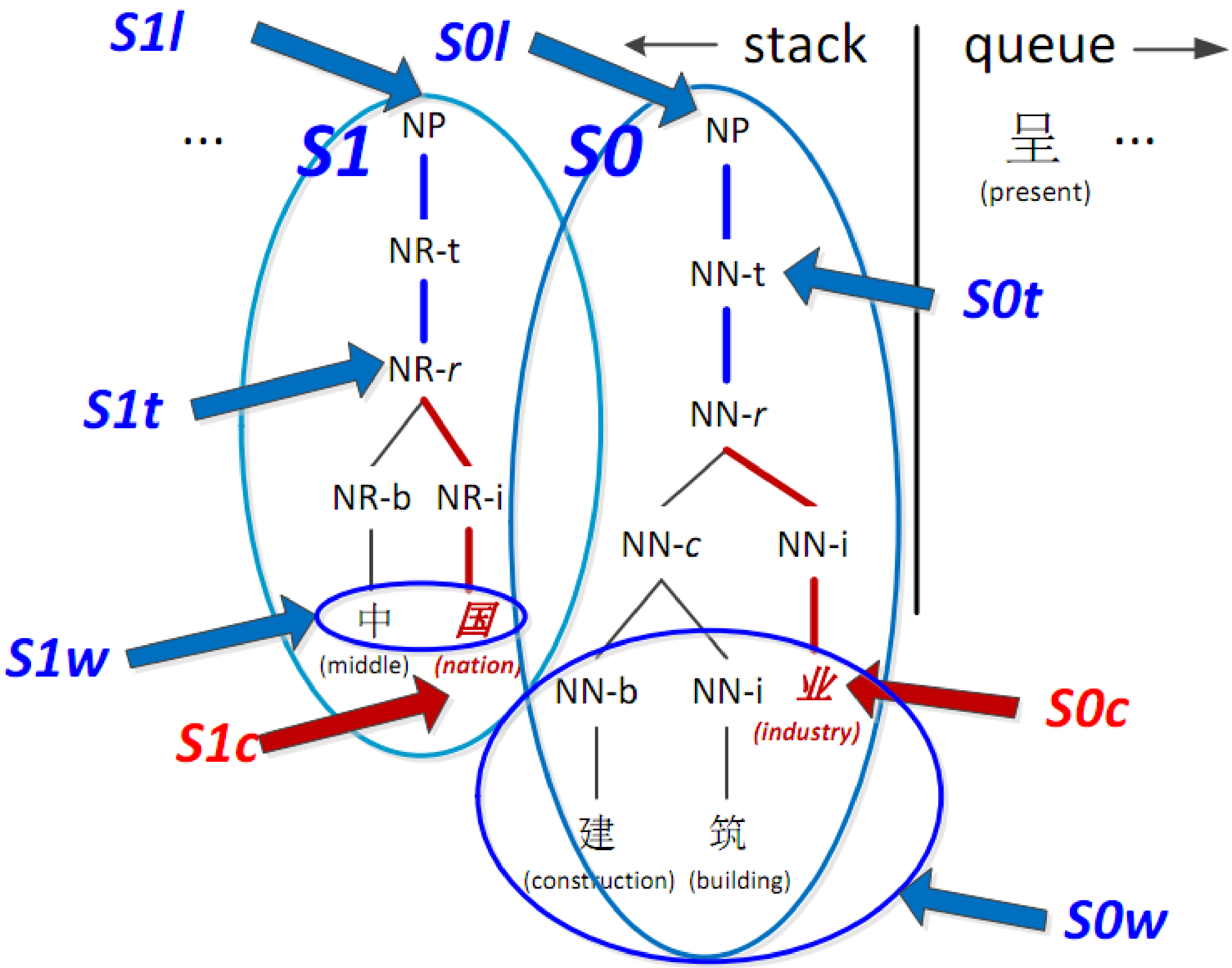
new features

Features



Zhang et al. ACL 2013

Features



Zhang et al. ACL 2013

Experiments

- Penn Chinese Treebank 5 (CTB-5)

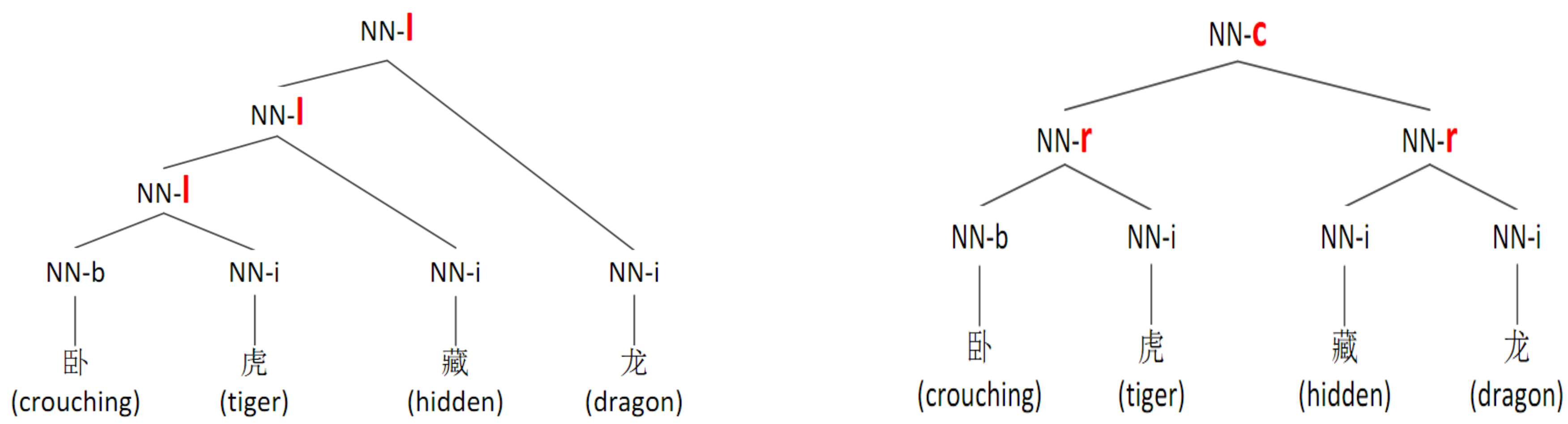
	CTB files	# sent.	# words
Training	1-270	18089	493,939
	400-1151		
Develop	301-325	350	6,821
Test	271-300	348	8,008

Experiments

- Baseline models
  - Pipeline model including:
    - Joint SEG&POS-Tagging model (Zhang and Clark, 2010).
    - Word-based CFG parsing model (Zhang and Clark, 2009).

Experiments

- Our proposed models
  - Joint model with flat word structures
  - Joint model with annotated word structures



Zhang et al. ACL 2013

Results

	Task	P	R	F
Pipeline	Seg	97.35	98.02	97.69
	Tag	93.51	94.15	93.83
	Parse	81.58	82.95	82.26
Flat word structures	Seg	97.32	98.13	97.73
	Tag	94.09	94.88	94.48
	Parse	83.39	83.84	83.61
Annotated word structures	Seg	97.49	98.18	97.84
	Tag	94.46	95.14	94.80
	Parse	84.42	84.43	84.43
	WS	94.02	94.69	94.35

Zhang et al. ACL 2013

Compare with other systems

Task	Seg	Tag	Parse
Kruengkrai+ ’09	97.87	93.67	–
Sun ’11	98.17	94.02	–
Wang+ ’11	98.11	94.18	–
Li ’11	97.3	93.5	79.7
Li+ ’12	97.50	93.31	–
Hatori+ ’12	98.26	94.64	–
Qian+ ’12	97.96	93.81	82.85
Ours pipeline	97.69	93.83	82.26
Ours joint flat	97.73	94.48	83.61
Ours joint annotated	97.84	94.80	84.43

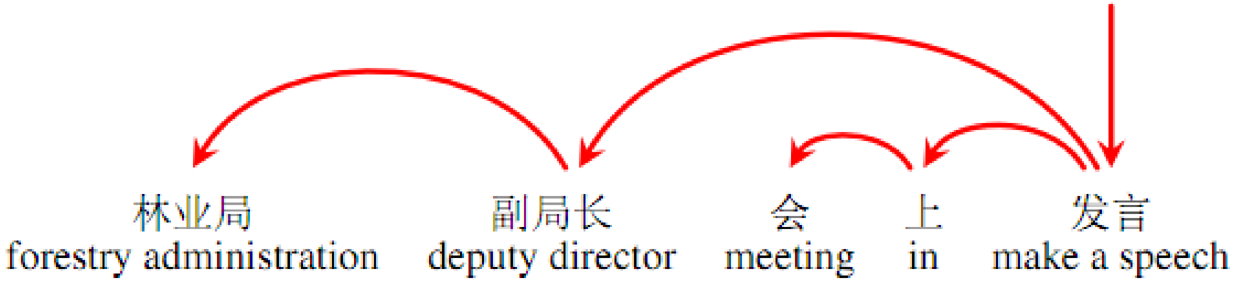
Zhang et al. ACL 2013

Applications

- **Word segmentation**
- **Dependency parsing**
- **Context free grammar parsing**
- **Combinatory categorial grammar parsing**
- **Joint segmentation and POS-tagging**
- **Joint POS-tagging and dependency parsing**
- **Joint segmentation, POS-tagging and constituent parsing**
- **Joint segmentation, POS-tagging and dependency parsing**

Traditional word-based dependency parsing

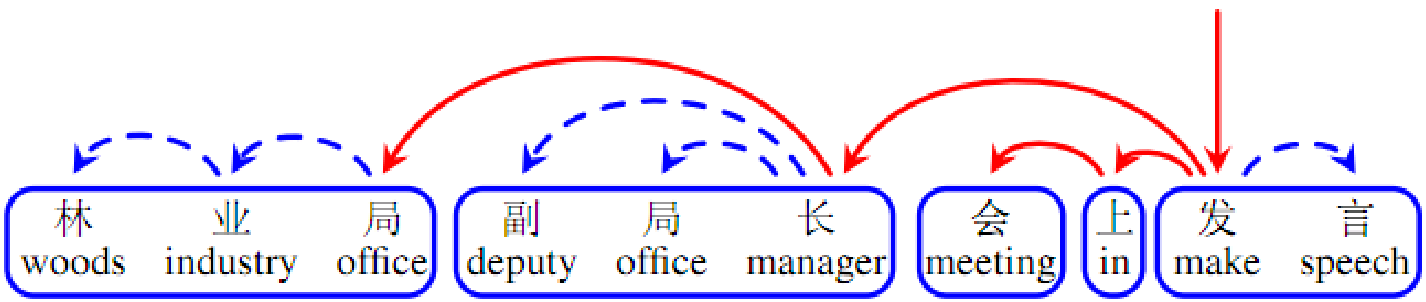
- Inter-word dependencies



Zhang et al. ACL 2014

Character-level dependency parsing

- Inter- and intra-word dependencies



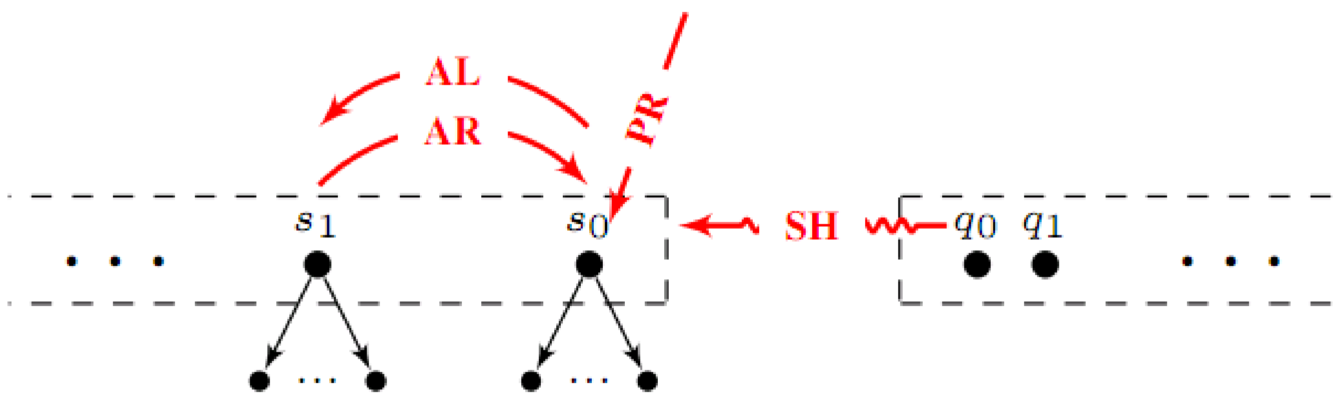
Zhang et al. ACL 2014

Main method

- An overview
  - Transition-based framework with global learning and beam search (Zhang and Clark, 2011)
  - Extensions from word-level transition-based dependency parsing models
    - Arc-standard (Nirve 2008; Huang et al., 2009)
    - Arc-eager (Nirve 2008; Zhang and Clark, 2008)

Main method

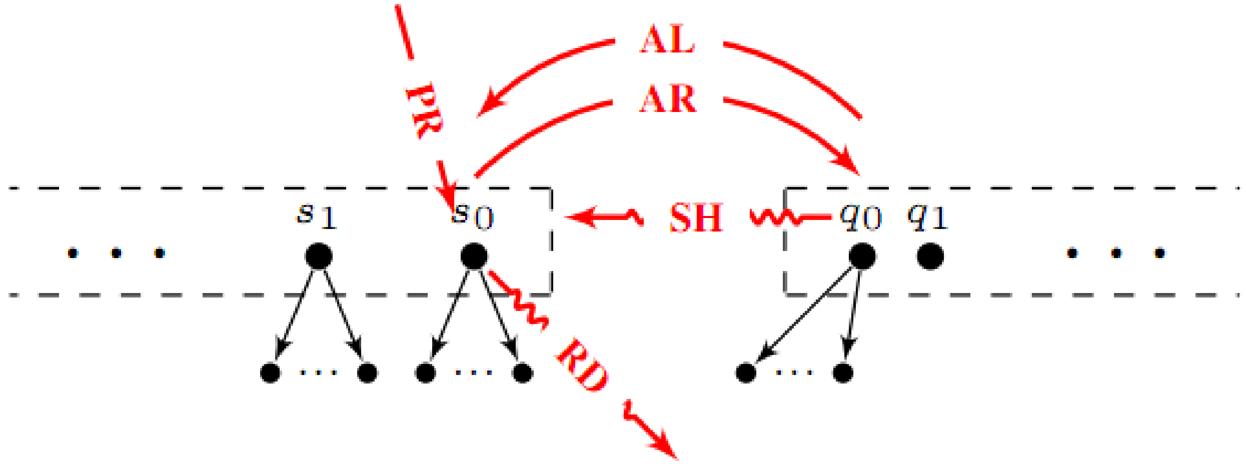
- Word-level transition-based dependency parsing
  - Arc-standard



Zhang et al. ACL 2014

Main method

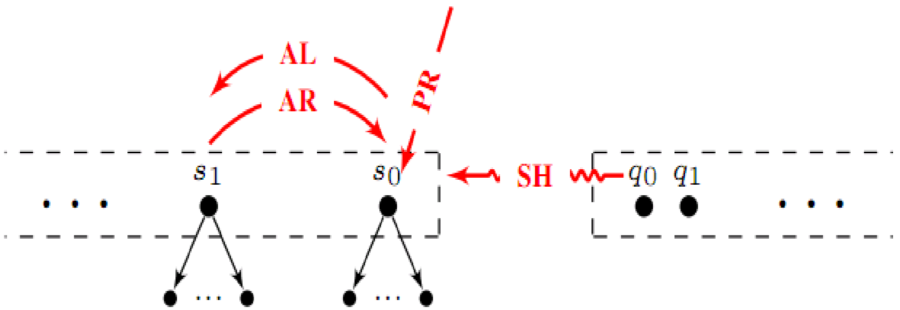
- Word-level transition-based dependency parsing
  - Arc-eager

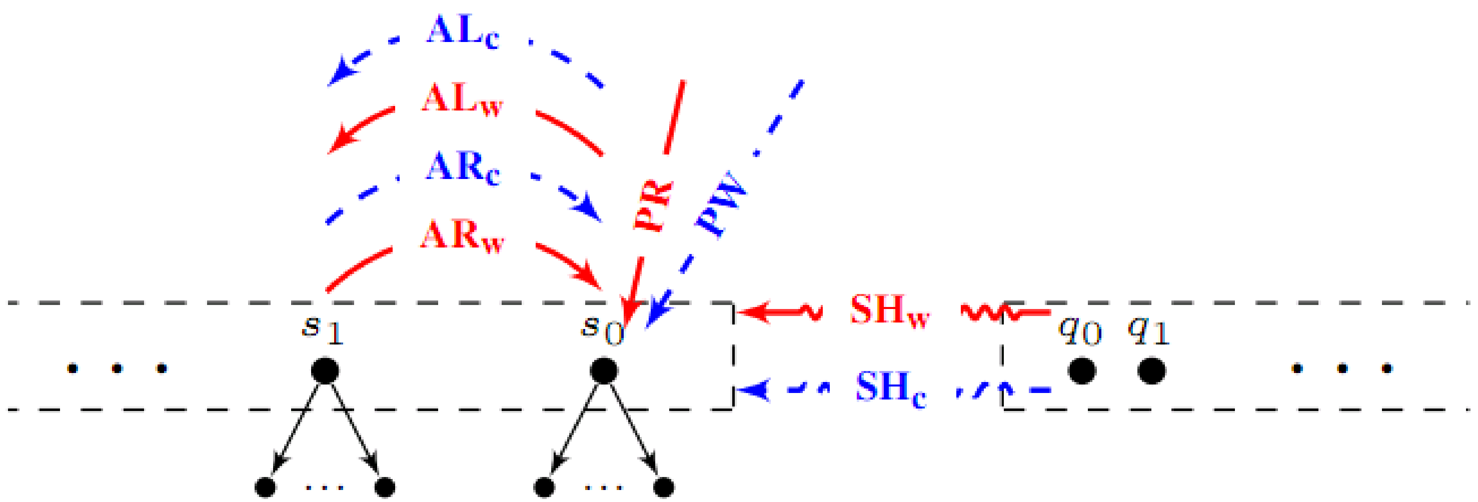


Zhang et al. ACL 2014

Main method

- Word-level to character-level
  - Arc-standard





Zhang et al. ACL 2014

Main method

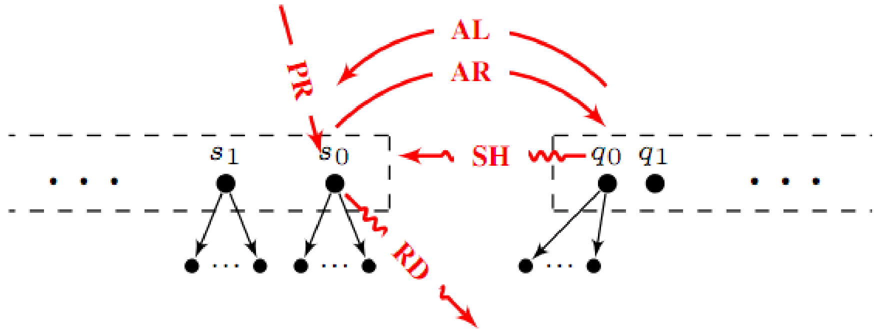
- Word-level to character-level
  - Arc-standard

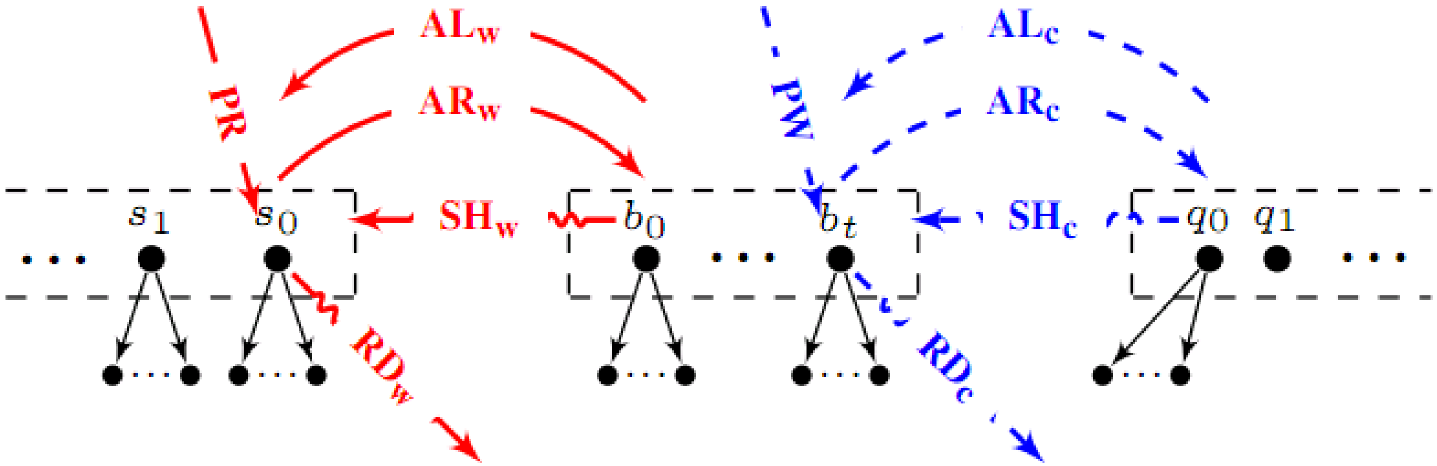
step	action	stack	queue	dependencies
0	-	ϕ	林 业 ···	ϕ
1	$SH_w(NR)$	林/NR	业 局 ···	ϕ
2	SH_c	林/NR 业/NR	局 副 ···	ϕ
3	AL_c	业/NR	局 副 ···	$A_1 = \{林 \curvearrowleft 业\}$
4	SH_c	业/NR 局/NR	副 局 ···	A_1
5	AL_c	局/NR	副 局 ···	$A_2 = A_1 \bigcup \{业 \curvearrowleft 局\}$
6	PW	林业局/NR	副 局 ···	A_2
7	$SH_w(NN)$	林业局/NR 副/NN	局 长 ···	A_2
···	···	···	···	···
12	PW	林业局/NR 副局长/NN	会 上 ···	A_i
13	AL_w	副局长/NN	会 上 ···	$A_{i+1} = A_i \bigcup \{林业局/NR \curvearrowleft 副局长/NN\}$
···	···	···	···	···

Zhang et al. ACL 2014

Main method

- Word-level to character-level
  - Arc-eager





Main method

- Word-level to character-level
  - Arc-eager

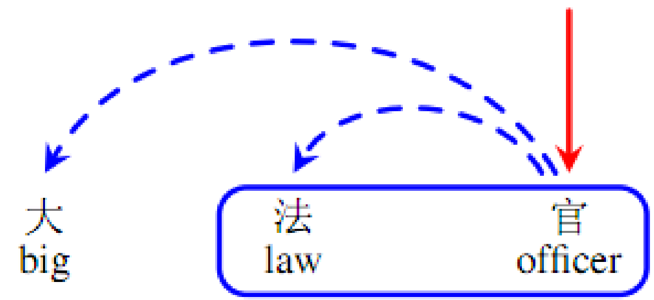
step	action	stack	deque	queue	dependencies
0	-	ϕ		林 业 ···	
1	SH_c(NR)	ϕ	林/NR	业 局 ···	ϕ
2	AL_c	ϕ	ϕ	业/NR 局 ···	$A_1 = \{$林$\curvearrowleft$业$\}$
3	SH_c	ϕ	业/NR	局 副 ···	A_1
4	AL_c	ϕ	ϕ	局/NR 副 ···	$A_2 = A_1 \bigcup \{$业$\curvearrowleft$局$\}$
5	SH_c	ϕ	局/NR	副 局 ···	A_2
6	PW	ϕ	林业局/NR	副 局 ···	A_2
7	SH_w	林业局/NR	ϕ	副 局 ···	A_2
···	···	···	···	···	···
13	PW	林业局/NR	副局长/NN	会 上 ···	A_i
14	AL_w	ϕ	副局长/NN	会 上 ···	$A_{i+1} = A_i \bigcup \{$林业局/NR$\curvearrowleft$副局长/NN$\}$
···	···	···	···	···	···

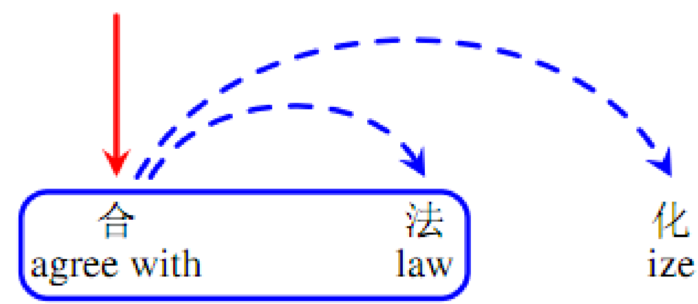
Zhang et al. ACL 2014

Main method

- New features

Feature templates
$L\underline{c}$, $L\underline{c}\underline{t}$, $R\underline{c}$, $R\underline{c}\underline{t}$, $L_{lc1}\underline{c}$, $L_{rc1}\underline{c}$, $R_{lc1}\underline{c}$,
$L\underline{c} \cdot R\underline{c}$, $L_{lc1}\underline{c}\underline{t}$, $L_{rc1}\underline{c}\underline{t}$, $R_{lc1}\underline{c}\underline{t}$,
$L\underline{c} \cdot R\underline{w}$, $L\underline{w} \cdot R\underline{c}$, $L\underline{c}\underline{t} \cdot R\underline{w}$,
$L\underline{w}\underline{t} \cdot R\underline{c}$, $L\underline{w} \cdot R\underline{c}\underline{t}$, $L\underline{c} \cdot R\underline{w}\underline{t}$,
$L\underline{c} \cdot R\underline{c} \cdot L_{lc1}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot L_{rc1}\underline{c}$,
$L\underline{c} \cdot R\underline{c} \cdot L_{lc2}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot L_{rc2}\underline{c}$,
$L\underline{c} \cdot R\underline{c} \cdot R_{lc1}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot R_{lc2}\underline{c}$,
$L\underline{lsw}$, $L\underline{rsw}$, $R\underline{lsw}$, $R\underline{rsw}$, $L\underline{lsw}\underline{t}$,
$L\underline{rsw}\underline{t}$, $R\underline{lsw}\underline{t}$, $R\underline{rsw}\underline{t}$, $L\underline{lsw} \cdot R\underline{w}$,
$L\underline{rsw} \cdot R\underline{w}$, $L\underline{w} \cdot R\underline{lsw}$, $L\underline{w} \cdot R\underline{rsw}$





Zhang et al. ACL 2014

Experiments

- Data
  - CTB5.0, CTB6.0, CTB7.0

		CTB50	CTB60	CTB70
Training	#sent	18k	23k	31k
	#word	494k	641k	718k
Development	#sent	350	2.1k	10k
	#word	6.8k	60k	237k
	#oov	553	3.3k	13k
Test	#sent	348	2.8k	10k
	#word	8.0k	82k	245k
	#oov	278	4.6k	13k

Zhang et al. ACL 2014

Experiments

- Proposed models
  - STD (real, pseudo)
    - Joint segmentation and POS-tagging with inner dependencies
  - STD (pseudo, real)
    - Joint segmentation, POS-tagging and dependency parsing
  - STD (real, real)
    - Joint segmentation, POS-tagging and dependency parsing with inner dependencies

  - EAG (real, pseudo)
    - Joint segmentation and POS-tagging with inner dependencies
  - EAG (pseudo, real)
    - Joint segmentation, POS-tagging and dependency parsing
  - EAG (real, real)
    - Joint segmentation, POS-tagging and dependency parsing with inner dependencies

Zhang et al. ACL 2014

Experiments

- Final results

Model	CTB50				CTB60				CTB70			
	SEG	POS	DEP	WS	SEG	POS	DEP	WS	SEG	POS	DEP	WS
The arc-standard models												
STD (pipe)	97.53	93.28	79.72	–	95.32	90.65	75.35	–	95.23	89.92	73.93	–
STD (real, pseudo)	97.78	93.74	–	**97.40**	**95.77**‡	91.24‡	–	**95.08**	**95.59**‡	90.49‡	–	**94.97**
STD (pseudo, real)	97.67	94.28‡	81.63‡	–	95.63‡	**91.40**‡	76.75‡	–	95.53‡	90.75‡	75.63‡	–
STD (real, real)	**97.84**	**94.62**‡	**82.14**‡	97.30	95.56‡	91.39‡	**77.09**‡	94.80	95.51‡	**90.76**‡	**75.70**‡	94.78
Hatori+ '12	97.75	94.33	81.56	–	95.26	91.06	75.93	–	95.27	90.53	74.73	–
The arc-eager models												
EAG (pipe)	97.53	93.28	79.59	–	95.32	90.65	74.98	–	95.23	89.92	73.46	–
EAG (real, pseudo)	97.75	93.88	–	97.45	95.63‡	91.07‡	–	95.06	**95.50**‡	90.36‡	–	**95.00**
EAG (pseudo, real)	97.76	**94.36**‡	81.70‡	–	95.63‡	91.34‡	76.87‡	–	95.39‡	90.56‡	75.56‡	–
EAG (real, real)	**97.84**	**94.36**‡	**82.07**‡	**97.49**	**95.71**‡	**91.51**‡	**76.99**‡	**95.16**	95.47‡	**90.72**‡	**75.76**‡	94.94

Zhang et al. ACL 2014

Experiments

- Analysis: word structure predication
  - OOV words
    - Overall

STD(real,real)	67.98%
EAG(real,real)	69.01%

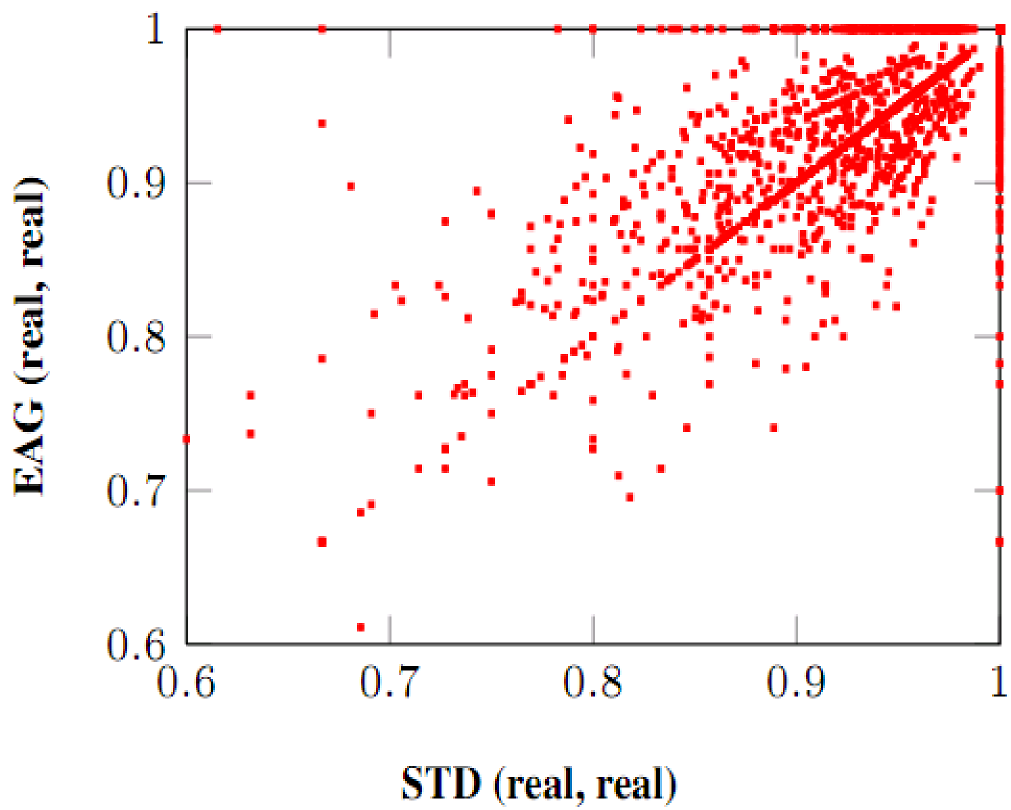
    - Assuming that the segmentation is correct

STD(real,real)	87.64%
EAG(real,real)	89.07%

Zhang et al. ACL 2014

Experiments

- Analysis: word structure predication
  - OOV words



Zhang et al. ACL 2014

Outline

Introduction	Applications
Analysis	ZPar

Analysis

- **Empirical analysis**
- **Theoretical analysis**

Analysis

- **Empirical analysis**
- **Theoretical analysis**

Empirical analysis

- Effective on all the tasks: beam-search + global learning + rich features
- What are the effects of global learning and beam-search, respectively
- Study empirically using dependency parsing

Empirical analysis

- Learning, search, features
  - Arc-eager parser
  - Learning
    - Global training
      - Optimize the entire transition sequence for a sentence
      - Structured predication
    - Local training
      - Each transition is considered in isolation
      - No global view of the transition sequence for a sentence
      - Classfier

Zhang and Nivre, COLING 2012

Empirical analysis

- Learning, search, features
  - Arc-eager parser
  - Learning
  - Features
    - Base features (local features) (Zhang and Clark, EMNLP 2008)
      - Features refer to combinations of atomic features (words and their POS tags) of the nodes on the stack and in the queue only.
    - All features (including rich non-local features) (Zhang and Nirve, ACL 2011)
      - Dependency distance
      - Valence
      - Grand and child features
      - Third-order features

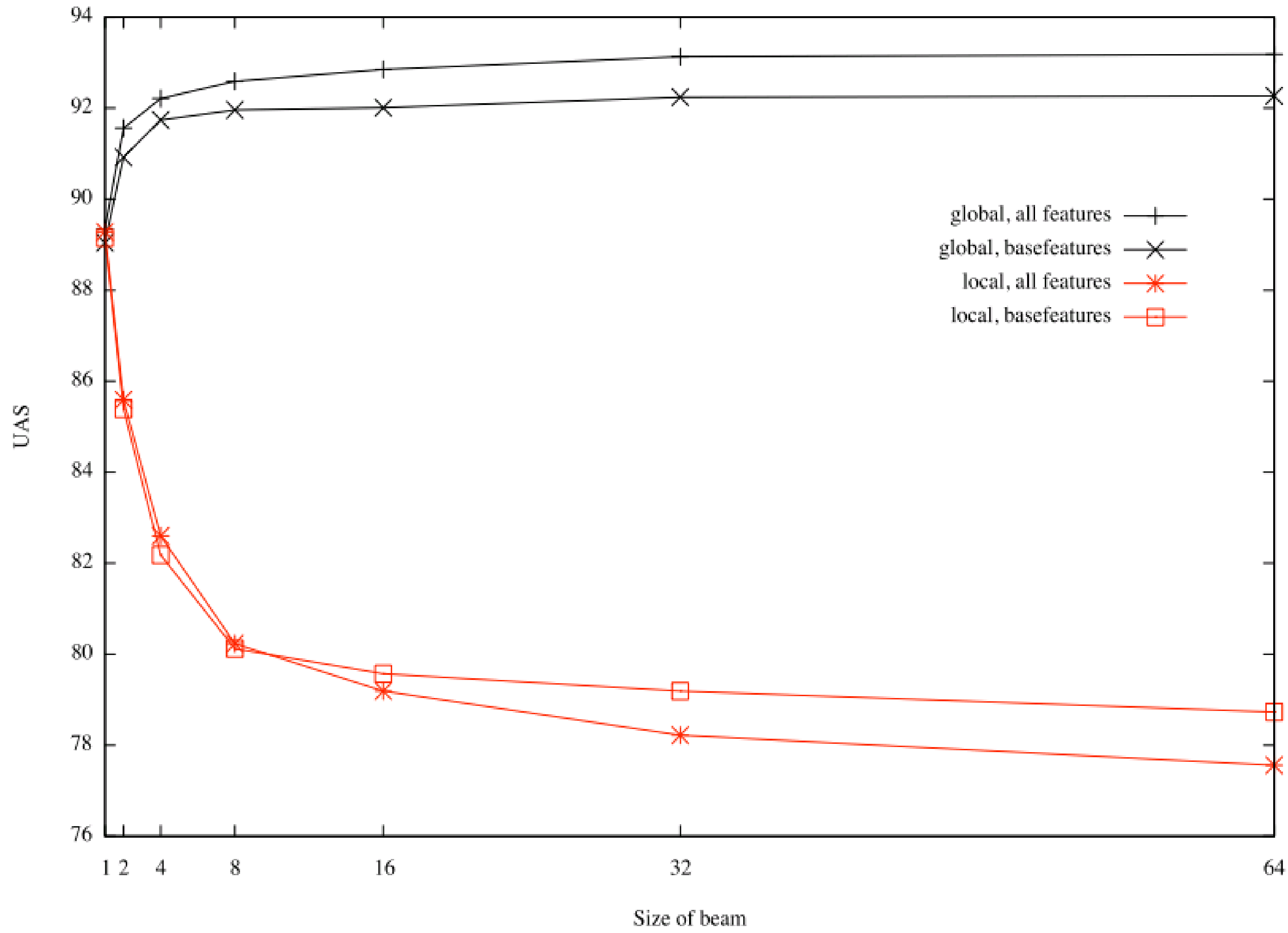
Empirical analysis

- Learning, search, features
  - Arc-eager parser
  - Learning
  - Features
  - Search
    - Beam = 1, greedy
    - Beam > 1

Zhang and Nivre, COLING 2012

Empirical analysis

- Contrast



Zhang and Nivre, COLING 2012

Empirical analysis

- Observations
  - Beam = 1, global learning ≈ local learning
  - Beam > 1, global learning ↑, local learning ↓
  - Richer features, make ↑ or ↓ faster.

Empirical analysis

- Why does not local learning benefit from beam-search?

training beam	testing beam	UAS
1	1	89.04
1	64	79.34
64	1	87.07
64	64	92.27

Zhang and Nivre, COLING 2012

Empirical analysis

- Does greedy, local learning benefit from rich features?
- Beam search (Zpar) and Greedy search (Malt) with non-local features

	ZPar	Malt
Baseline	92.18	89.37
+distance	+0.07	−0.14
+valency	+0.24	0.00
+unigrams	+0.40	−0.29
+third-order	+0.18	0.00
+label set	+0.07	+0.06
Extended	93.14	89.00

Zhang and Nivre, COLING 2012

Empirical analysis

- Conclusions
  - Global learning and beam-search benefit each other
  - Global learning and beam-search accommodate richer features without overfitting
  - Global learning and beam-search should be used simultaneously

Analysis

- **Empirical analysis**
- **Theoretical analysis**

Theoretical analysis

- The perceptron
  - Online learning framework

Inputs : training ex…les $(x_i, y_i)|_{i=1}^{T}$

Initialization : set $\vec{w}$

Algorithm :

for $r = 1 \cdots$

for $i = 1 \cdots$

calculate $z_i = \text{decode}(w, x_i)$

if($z_i \neq$ …)

$\vec{w}$ … $\vec{w}$ … $(\ldots, \ldots_i) - \phi(x_i, z_i)$

output : $\vec{w}$

Michael Collins, EMNLP 2002

Theoretical analysis

- The perceptron
  - If the data $(x_t, y_t)|_{t=1}^{T}$ is separable and for all $\|\phi(x,y)\| \leq R$, then there exists some $\lambda > 0$, making the max error number (updating number) be less than R^2/λ^2

$$w^{k+1}u = (w^k + (\phi(x_t, y_t) - \phi(x_t, y^p)))u$$

$$= w^k u + (\phi(x_t, y_t) - \phi(x_t, y^p))u$$

if u can seperate the data, then

$$\phi(x_t, y_t)u > \phi(x_t, y^p))u$$

thus, $w^{k+1}u \geq w^k u + \lambda$

assume $w^0 = 0$ and another fact $\|u\| = 1$,

then $w^{k+1} \geq k\lambda$

Michael Collins, EMNLP 2002

Theoretical analysis

- The perceptron
  - If the data $(x_t, y_t)|_{t=1}^{T}$ is separable and for all $\|\phi(x, y)\| \le R$, then there exists some $\lambda > 0$, making the max error number (updating number) be less than R^2/λ^2

$$w^{k+1}u = (w^k + (\phi(x_t, y_t) - \phi(x_t, y^p)))u$$

$$= w^k u + (\phi(x_t, y_t) - \phi(x_t, y^p))u$$

if u can seperate the data, then

$$\phi(x_t, y_t)u > \phi(x_t, y^p))u$$

the margin

thus, $w^{k+1}u \ge w^k u + \lambda$

assume $w^0 = 0$ and another fact $||u|| = 1$,

then $w^{k+1} \ge k\lambda$

Michael Collins, EMNLP 2002

Theoretical analysis

- The perceptron
  - If the data $(x_t, y_t)|_{t=1}^{T}$ is separable and for all $\|\phi(x, y)\| \leq R$, then there exists some $\lambda > 0$, making the max error number (updating number) be less than R^2/λ^2

$$w^{k+1} = w^{k+1} + (\phi(x_t, y_t) - \phi(x_t, y^p))$$

$$\|w^{k+1}\|^2 = \|w^k\|^2 + 2(\phi(x_t, y_t) - \phi(x_t, y^p))w^k + \|\phi(x_t, y_t) - \phi(x_t, y^p)\|^2$$

if we have this update, then

$$\phi(x_t, y_t)w^k < \phi(x_t, y^p))w^k$$

thus, $\|w^{k+1}\|^2 \leq \|w^k\|^2 + \|\phi(x_t, y_t) - \phi(x_t, y^p)\|^2 \leq \|w^k\|^2 + 4R^2$

assume $w^0 = 0$

then $\|w^{k+1}\|^2 \leq 4kR^2$

Michael Collins, EMNLP 2002

Theoretical analysis

- The perceptron
  - If the data $(x_t, y_t)|_{t=1}^{T}$ is separable and for all $\|\phi(x, y)\| \leq R$, then there exists some $\lambda > 0$, making the max error number (updating number) be less than R^2/λ^2

$$w^{k+1} = w^{k+1} + (\phi(x_t, y_t) - \phi(x_t, y^p))$$

$$\|w^{k+1}\|^2 = \|w^k\|^2 + 2(\phi(x_t, y_t) - \phi(x_t, y^p))w^k + \|\phi(x_t, y_t) - \phi(x_t, y^p)\|^2$$

if we have this update, then

$$\phi(x_t, y_t)w^k < \phi(x_t, y^p))w^k$$

This is satisfied in dynamic programming, it may not hold in beam-search

thus, $\|w^{k+1}\|^2 \leq \|w^k\|^2 + \|\phi(x_t, y_t) - \phi(x_t, y^p)\|^2 \leq \|w^k\|^2 + 4R^2$

assume $w^0 = 0$

then $\|w^{k+1}\|^2 \leq 4kR^2$

Michael Collins, EMNLP 2002

Theoretical analysis

- The perceptron
  - If the data $(x_t, y_t)|_{t=1}^{T}$ is separable and for all $\|\phi(x,y)\| \leq R$, then there exists some $\lambda > 0$, making the max error number (updating number) be less than R^2/λ^2

$$w^{k+1} \geq k\lambda$$

$$\|w^{k+1}\|^2 \leq 4kR^2$$

$$\text{Thus, } k^2\lambda^2 \leq \|w^{k+1}\|^2 \leq 4kR^2$$

$$k \leq \frac{4R^2}{\lambda^2}, \text{ another words, also } k \leq \frac{R^2}{\lambda^2}$$

Michael Collins, EMNLP 2002

Theoretical analysis

- The perceptron
  - If the data $(x_t, y_t)|_{t=1}^{T}$ is not separable, we should assume that there is an oracle **u** so that the number of errors made by it is o(T).

$$w^{k+1}u = (w^k + (\phi(x_t, y_t) - \phi(x_t, y^p)))u$$

$$= w^k u + (\phi(x_t, y_t) - \phi(x_t, y^p))u$$

thus when $k = CT$,

$$w^{k+1}u \geq (k - o(k))\lambda - o(k)CR + w^0 u \geq k\lambda - o(k) + w^0 u$$

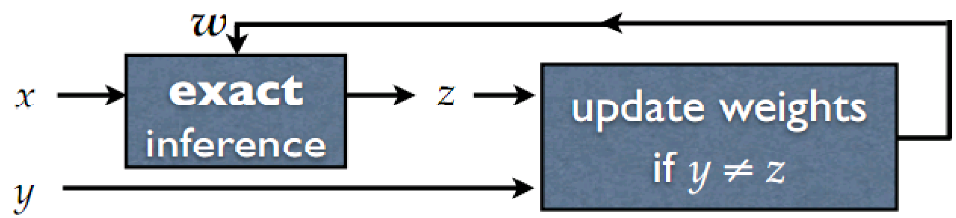
assume $w^0 = 0$ and another fact $||u|| = 1$,

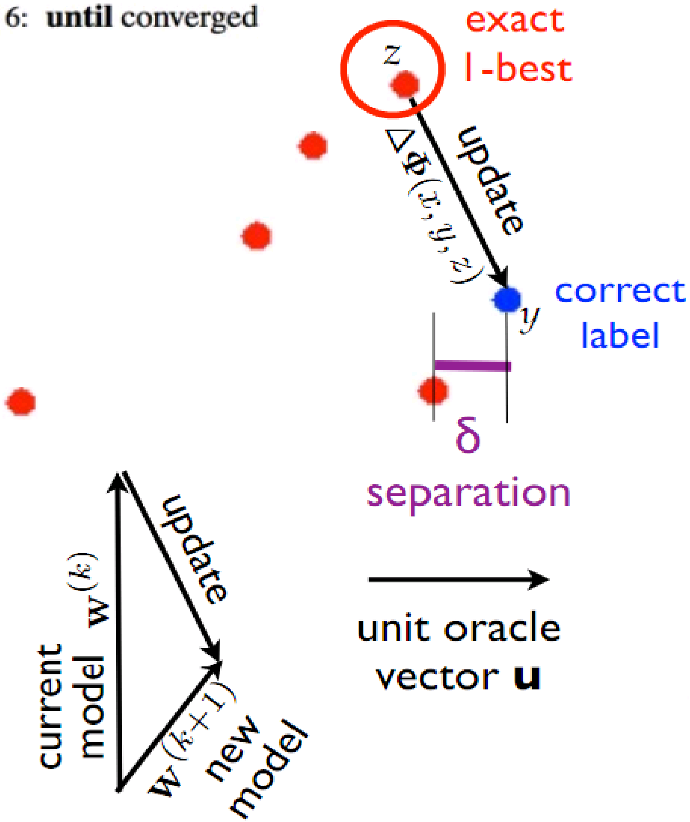
then $w^{k+1} \geq k\lambda - o(k)$

Theoretical analysis

- The perceptron

repeat
  for each example (x, y) **in** D **do**
    $z \leftarrow \textsc{Exact}(x, \mathbf{w})$
    if $z \neq y$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y, z)$
until converged





perceptron update:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta\mathbf{\Phi}(x, y, z)$$

$$\mathbf{u} \cdot \mathbf{w}^{(k+1)} = \mathbf{u} \cdot \mathbf{w}^{(k)} + \underbrace{\mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z)}_{\geq \delta \quad \text{margin}}$$

$$\mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta \qquad \text{(by induction)}$$

$$\|\mathbf{u}\|\|\mathbf{w}^{(k+1)}\| \geq \mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta$$

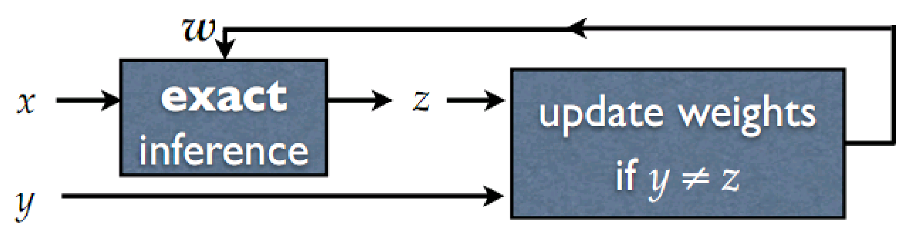
$$\|\mathbf{w}^{(k+1)}\| \geq k\delta$$

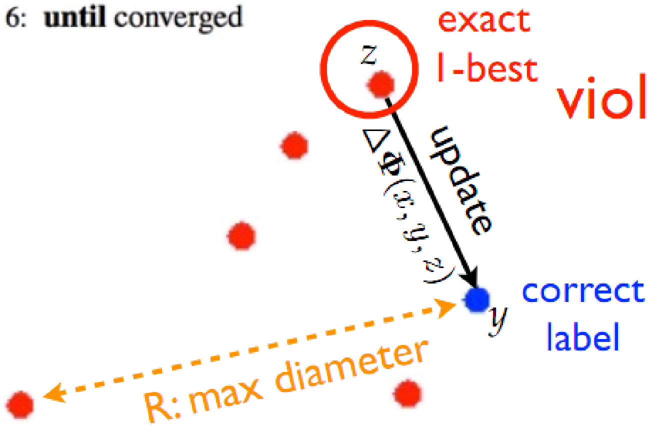
Huang et al., NAACL 2012

Theoretical analysis

- The perceptron

repeat
  for each example (x, y) in D **do**
    $z \leftarrow \textsc{Exact}(x, \mathbf{w})$
    if $z \neq y$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \Delta\Phi(x, y, z)$
until converged





violation: incorrect label scored higher

perceptron update:

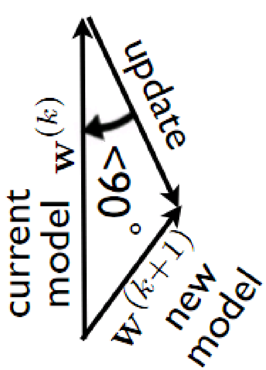
$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta\Phi(x, y, z)$$

$$\|\mathbf{w}^{(k+1)}\|^2 = \|\mathbf{w}^{(k)} + \Delta\Phi(x, y, z)\|^2$$

$$= \|\mathbf{w}^{(k)}\|^2 + \|\Delta\Phi(x, y, z)\|^2 + 2\, \mathbf{w}^{(k)} \cdot \Delta\Phi(x, y, z)$$

$\|\Delta\Phi(x, y, z)\|^2 \leq R^2$ diameter

$2\, \mathbf{w}^{(k)} \cdot \Delta\Phi(x, y, z) \leq 0$ violation



by induction: $\|\mathbf{w}^{(k+1)}\|^2 \leq kR^2$

Huang et al., NAACL 2012

Theoretical analysis

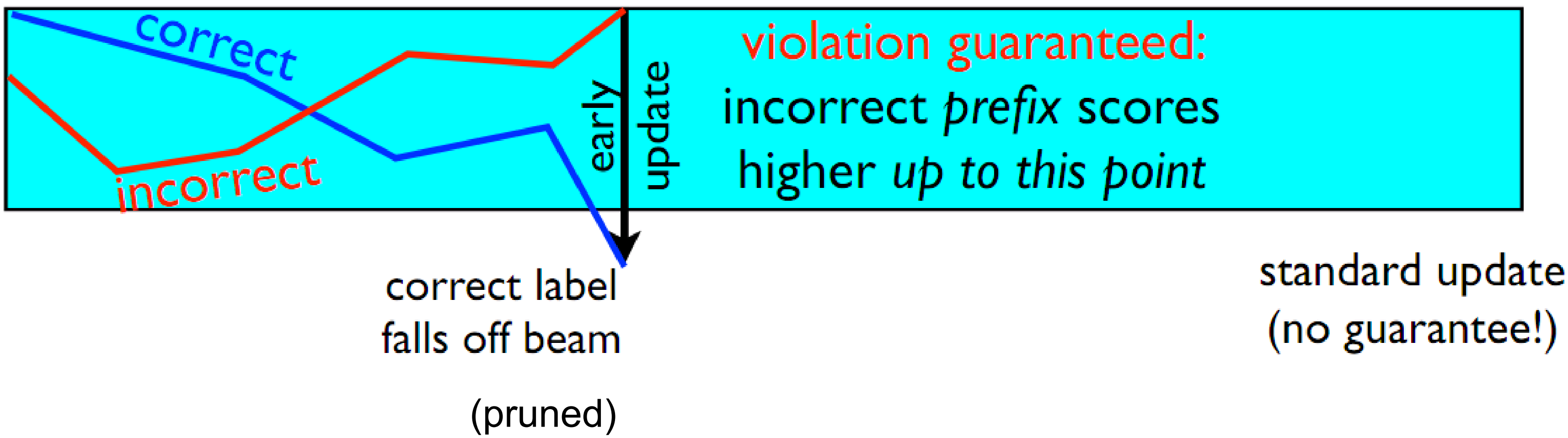
- The perceptron
  - The third factor must be less than zero! (violation)

$$\|\mathbf{w}^{(k)}\|^2 + \underbrace{\|\Delta\Phi(x,y,z)\|^2}_{\leq R^2 \text{ (diameter)}} + \underbrace{2\,\mathbf{w}^{(k)} \cdot \Delta\Phi(x,y,z)}_{\leq 0 \text{ (violation)}}$$

Huang et al., NAACL 2012

Theoretical analysis

- Why early-update?
  - early update -- when correct label first falls off the beam
    - up to this point the incorrect prefix should score higher
  - standard update (full update) -- no guarantee!



Huang et al., NAACL 2012

Outline

Introduction	Applications
Analysis	**ZPar**

ZPar

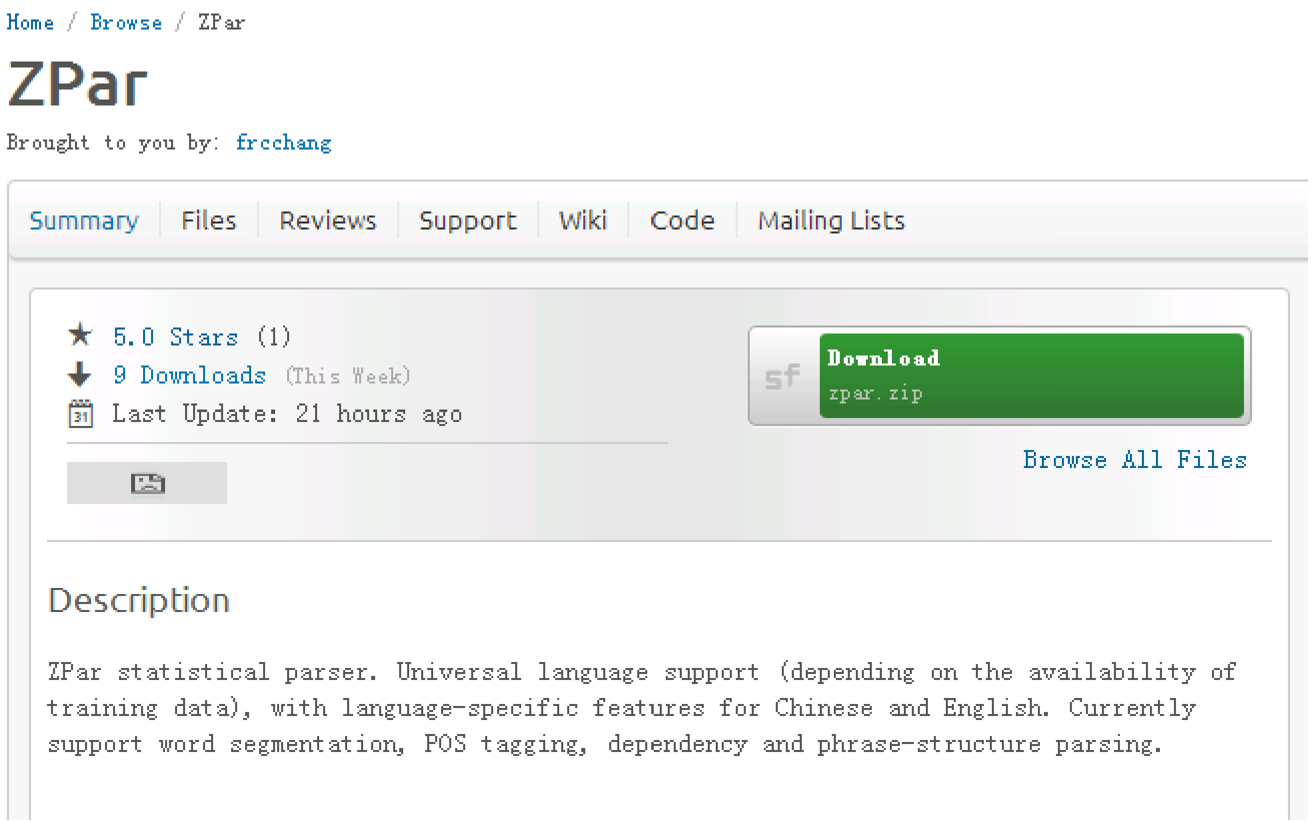
- Introduction
- Usage
- Development
- On-going work
- Contributions welcome

ZPar

- **Brief introduction**
- **Usage**
- **Development**
- **On-going work**
- **Contributions welcome**

Brief introduction

- Initiated in 2009 at Oxford, extended at Cambridge and SUTD, with more developers being involved



Brief introduction

- 2009—2014, Oxford, Cambridge, SUTD
- Functionalities extended

Categories

Features

- Chinese word segmentor
- Chinese and English pos tagger
- Chinese and English dependency parser
- Chinese and English constituent parser
- Multiple language parsers
- Chinese sentence boundary separator
- Statistical NLP tools

Brief introduction

- 2009—2014, Oxford, Cambridge, SUTD
- Functionalities extended
- Released several versions

Name	Modified	Size
0.6	2013-09-17	
0.5	2011-11-18	
0.4	2010-09-27	
0.3	2010-04-16	
0.2	2010-03-23	
0.1	2009-09-28	

Brief introduction

- 2009—2014, Oxford, Cambridge, SUTD
- Functionalities extended
- Released several versions
- Contains all implementations of this tutorial
  - Segmentation
  - POS tagging (single or joint)
  - Dependency parsing (single or joint)
  - Constituent parsing (single or joint)
  - CCG parsing (single or joint)

Brief introduction

- 2009—2014, Oxford, Cambridge, SUTD
- Functionalities extended
- Released several versions
- Contains all implementations of this tutorial
- Code structure

```
src
|-- chinese
|   |-- cfg
|   |-- dependency
|   |-- doc2snt
|   |-- joint
|   |-- pos
|   |-- segmentor
|   `-- tagger
|-- common
|   |-- conparser
|   |-- deplabeler
|   |-- depparser
|   `-- tagger
|-- english
|   |-- cfg
|   |-- dependency
|   `-- pos
|-- generic
|   `-- dependency
|-- include
|   |-- knowledge
|   |-- learning
|   `-- linguistics
`-- libs
    `-- linguistics
```

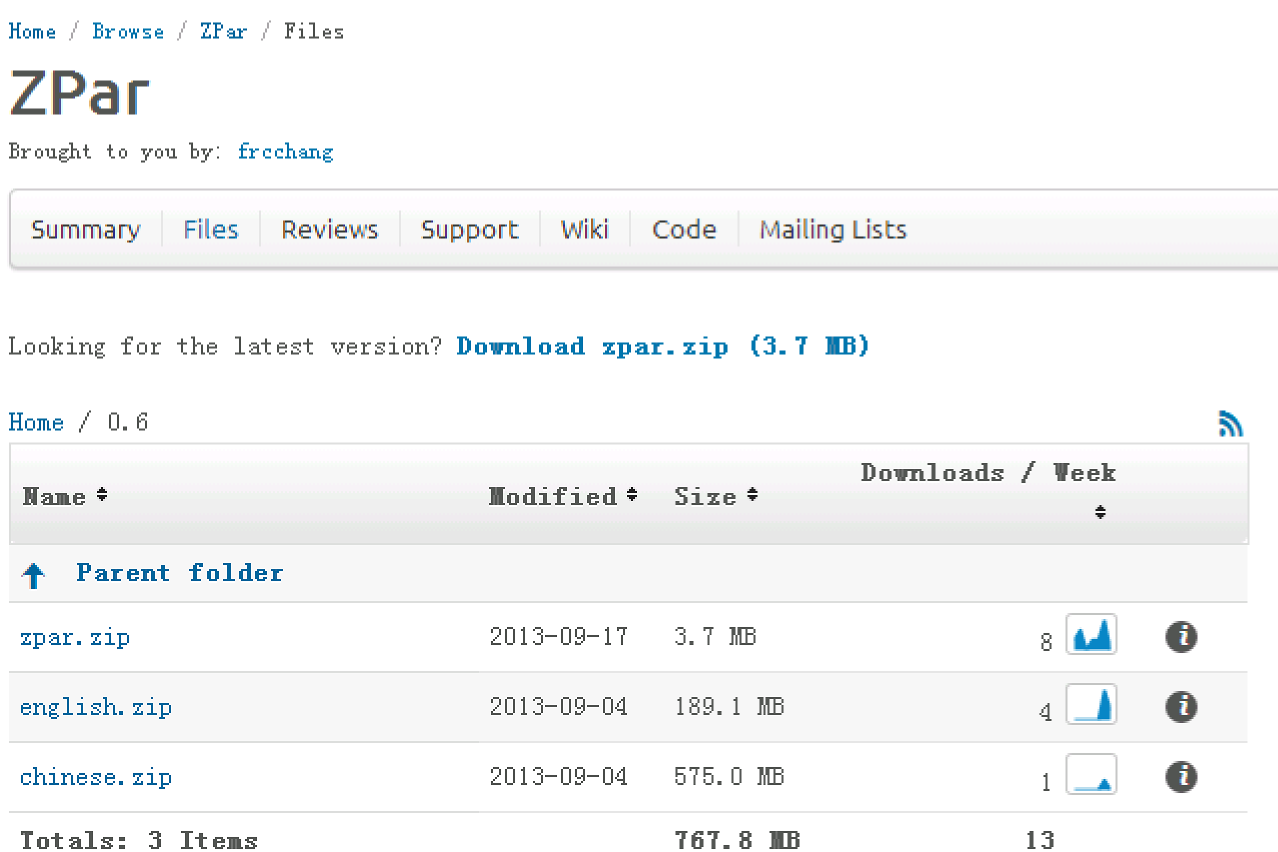
ZPar

- **Introduction**
- **Usage**
- **Development**
- **On-going work**
- **Contributions welcome**

Usage

- Download

  http://sourceforge.net/projects/zpar/files/0.6/



Home / Browse / ZPar / Files

ZPar

Brought to you by: frcchang

Summary | Files | Reviews | Support | Wiki | Code | Mailing Lists

Looking for the latest version? **Download zpar.zip (3.7 MB)**

Home / 0.6

Name	Modified	Size	Downloads / Week
Parent folder			
zpar.zip	2013-09-17	3.7 MB	8
english.zip	2013-09-04	189.1 MB	4
chinese.zip	2013-09-04	575.0 MB	1
Totals: 3 Items		**767.8 MB**	**13**

Usage

- For off-the-shelf Chinese language processing:
  - Compile: make zpar

```
[mszhang@node06:zpar]$ make zpar
mkdir -p ./obj
mkdir -p ./dist
g++ -W -O3 -I./src/include -DNDEBUG -I./src/chinese -c ./src/chinese/doc2snt/doc2snt.cpp -o ./obj/chinese.doc2snt.o
./src/chinese/charcat.h: In function 'int getStartingBracket(const CWord&)':
./src/chinese/charcat.h:70: warning: comparison between signed and unsigned integer expressions
mkdir -p ./obj
mkdir -p ./obj/linguistics
g++ -W -O3 -I./src/include -DNDEBUG -c src/libs/reader.cpp -o obj/reader.o
mkdir -p ./obj
mkdir -p ./obj/linguistics
g++ -W -O3 -I./src/include -DNDEBUG -c src/libs/writer.cpp -o obj/writer.o
```

```
./src/include/linguistics/cfgtemp.h:26: warning: base class 'class chinese::CConstituentLabel' should be
d in the copy constructor
g++  -o ./dist/zpar ./obj/zpar.o ./obj/chinese.postagger.o ./obj/chinese.postagger/weight.o ./obj/chinese.c
onparser.o ./obj/chinese.conparser/constituent.o ./obj/chinese.conparser/weight.o ./obj/chinese.depparser.o
 ./obj/chinese.depparser/weight.o ./obj/chinese.doc2snt.o ./obj/reader.o ./obj/writer.o ./obj/options.o ./o
bj/linguistics/lemma.o ./obj/linguistics/conll.o
The Chinese zpar system compiled successfully into ./dist.
```

Usage

- For off-the-shelf Chinese language processing:
  - Compile: make zpar
  - Usage

```
[mszhang@node06:zpar]$ cd dist/
[mszhang@node06:dist]$ ls
zpar
[mszhang@node06:dist]$ ./zpar

Usage: ./zpar feature_path [input_file [outout_file]]
Options:
 -o{s|t[d]|d|c}: outout format; 's' segmented format, 't' pos-tagged format in sentences, 'td' pos-tagged f
ormat in documents withstd::cout sentence boundary delimination, 'd' refers to dependency parse tree format
, and 'c' refers to constituent parse tree format. Default: c;
```

Usage

- For off-the-shelf Chinese language processing:
  - Compile: make zpar
  - Usage
  - Model download

Home / 0.6

Name	Modified	Size	Downloads / Week
Parent folder			
zpar.zip	2013-09-17	3.7 MB	8
english.zip	2013-09-04	189.1 MB	1
chinese.zip	2013-09-04	575.0 MB	1

Usage

- For off-the-shelf Chinese language processing:
  - Compile: make zpar
  - Usage
  - Model download
  - An example

```
[mszhang@node101:dist]$ tree ../chinese
./chinese
├─── conparser
├─── depparser
└─── tagger
```

```
[mszhang@node101:dist]$ ./zpar ../chinese -od
Initializing ZPar...
[The segmentation and tagging model] Loading scores ...set character knowledge...
 done. (8.94s)
[The parsing model] Loading scores... done. (10.06s)
ZPar initialized.
这是一个例子。
这      PN      1       SBJ
是      VC      -1      ROOT
一      CD      3       M
个      M       4       NMOD
例子    NN      1       VMOD
。      PU      1       VMOD
```

```
[mszhang@node101:dist]$ ./zpar ../chinese
Initializing ZPar...
[The parsing model] Loading scores... done. (125.43s)
ZPar initialized.
这是一个例子。
(IP (NP (PN#t (PN#b 这))) (VP (VC#t (VC#b 是)) (NP (QP (CD#t (CD#b 一)) (CLP (M#t (M#b 个)))) (NP (NN#t (NN#z (NN#b 例) (NN#i 子)))))) (PU#t (PU#b 。)))
```

Usage

- For off-the-shelf English language processing:
  - Compile: make zpar.en

```
[mszhang@node06:zpar]$ make zpar.en
mkdir -p ./obj
mkdir -p ./obj/linguistics
g++ -W -O3 -I./src/include -DNDEBUG -c src/libs/reader.cpp -o obj/reader.o
mkdir -p ./obj
mkdir -p ./obj/linguistics
g++ -W -O3 -I./src/include -DNDEBUG -c src/libs/writer.cpp -o obj/writer.o
src/libs/writer.cpp: In member function 'void CSentenceWriter::writeSentence(const CStringVector*, const s
td::string&, bool)' :
src/libs/writer.cpp:25: warning: comparison between signed and unsigned integer expressions
src/libs/writer.cpp: In member function 'void CSentenceWriter::writeSentence(const CTwoStringVector*, char
, bool)' :
```

```
./src/common/conparser/implementations/sr/rule.h:176:   instantiated from here
./src/include/linguistics/cfgtemp.h:26: warning: base class 'class english::CConstituentLabel'  should be
explicitly initialized in the copy constructor
g++  -o ./dist/zpar.en ./obj/zpar.en.o ./obj/english.postagger/weight.o ./obj/english.postagger.o ./obj/eng
lish.depparser.o ./obj/english.depparser/weight.o ./obj/english.conparser.o ./obj/english.conparser/constit
uent.o ./obj/english.conparser/weight.o ./obj/english.deplabeler.o ./obj/english.deplabeler/weight.o ./obj/
reader.o ./obj/writer.o ./obj/options.o ./obj/linguistics/lemma.o ./obj/linguistics/conll.o
The English zpar.en system compiled successfully into ./dist.
```

Usage

- For off-the-shelf English language processing:
  - Compile: make zpar.en
  - Usage

```
[mszhang@node06:zpar]$ cd dist/
[mszhang@node06:dist]$ ls
zpar.en
[mszhang@node06:dist]$ ./zpar.en

Usage: ./zpar.en feature_path [input_file [outout_file]]
Options:
 -o{t|d|c}: outout format; 't' pos-tagged format in sentences, 'd' refers to dependency parse tree format,
and 'c' refers to constituent parse tree format. Default: d;
```

Usage

- For off-the-shelf English language processing:
  - Compile: make zpar.en
  - Usage
  - Model download

Home / 0.6

Name	Modified	Size	Downloads / Week
Parent folder			
zpar.zip	2013-09-17	3.7 MB	8
english.zip	2013-09-04	189.1 MB	1
chinese.zip	2013-09-04	575.0 MB	1

Usage

- For off-the-shelf English language processing:
  - Compile: make zpar.en
  - Usage
  - Model download
  - An example

```
[mszhang@node06:dist]$ tree ../english
../english
|-- conparser
|-- depparser
`-- tagger
```

```
[mszhang@node06:dist]$ ./zpar.en ../english
Parsing started
[tagger] Loading model... done.
[parser] Loading scores... done. (23.1s)
ZPar is a parser .
ZPar    NNP     1       SUB
is      VBZ     -1      ROOT
a       DT      3       NMOD
parser  NN      1       PRD
.       .       1       P
```

```
[mszhang@node06:dist]$ ./zpar.en ../english -oc
Parsing started
[tagger] Loading model... done.
[parser] Loading scores... done. (59.59s)
ZPar is a parser .
(S (NP (NNP ZPar)) (VP (VBZ is) (NP (DT a) (NN parser))) (. .))
```

Usage

- A generic ZPar
  - For many languages the tasks are similar
  - POS-tagging (consists morphological analysis) and parsing

Usage

- For generic processing:
  - Compile: make zpar.ge
  - Usage

```
[mszhang@node06:zpar]$ cd dist/
[mszhang@node06:dist]$ ls
zpar.ge
[mszhang@node06:dist]$ ./zpar.ge

Usage: ./zpar.ge feature_path [input_file [outout_file]]
Options:
 -o{t|d|c}: outout format; 't' pos-tagged format in sentences, 'd' refers to labeled dependency tree format
, and 'c' refers to constituent parse tree format. Default: d;
```

Usage

- For generic processing:
  - Compile: make zpar.ge
  - Usage
  - An example

```
[mszhang@node06:dist]$ tree ../english
../english
|-- conparser
|-- depparser
`-- tagger
```

```
[mszhang@node06:dist]$ ./zpar.ge ../english
Parsing started
[POS tagging module] Loading model... done.
[Parsing module] Loading scores... done. (19.41s)
ZPar is a parser .
ZPar    NNP     1       SBAR
is      VBZ     -1      -ROOT-
a       DT      3       DEP
parser  NN      1       OBJ
.       .       1       OBJ
```

Usage

- Using the individual components
  - Chinese word segmentation
    - Makefile modification

```
SEGMENTOR_IMPL = agenda
```

    - Make

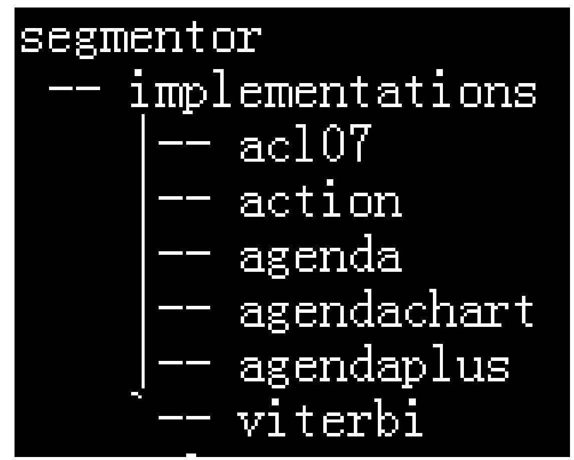
```
make segmentor
```

    - Train

```
./train input_file model_file iteration
```

    - Decode

```
./segmentor model_file input_file output_file
```



Usage

- Using the individual components
  - Chinese/English POS tagger
    - Makefile modification

```
CHINESE_TAGGER_IMPL = agenda
ENGLISH_TAGGER_IMPL = agenda
```

    - Make

```
make chinese.postagger
make english.postagger
```

    - Train

```
./train input_file model_file iteration
```

    - Decode

```
./tagger model_file input_file output_file
```

For English POS-tagging

```
tagger
`-- implementations
    |-- agenda
    `-- collins
```

For Chinese POS-tagging

```
tagger
`-- implementations
    |-- agenda
    |-- agendachart
    |-- agendanew
    |-- segmented
    `-- spa
```

Usage

- Using the individual components
  - Chinese/English dependency parsing
    - Makefile modification

```
CHINESE_DEPPARSER_IMPL = arceager
ENGLISH_DEPPARSER_IMPL = arceager
```

    - Make

```
make chinese.depparser
make english.depparser
```

    - Train

```
./train input_file model_file iteration
```

    - Decode

```
./tagger input_file output_file model_file
```

```
depparser
`-- implementations
    |-- acl11
    |-- arceager
    |-- cad
    |-- covington
    |-- eisner
    |-- emnlp08
    |-- meta
    |-- punct
    `-- uppsala
```

Usage

- Using the individual components
  - Chinese/English constituent parsing
    - Makefile modification

```
CHINESE_CONPARSER_IMPL = cad
ENGLISH_CONPARSER_IMPL = cad
```

    - Make

```
make chinese.conparser
make english.conparser
```

    - Train

```
./train input_file model_file iteration
```

    - Decode

```
./tagger input_file output_file model_file
```

For English/Chinese constituent parsing

```
conparser
`-- implementations
    |-- cad
    |-- enode
    |-- lin
    |-- muhua
    `-- srnew
```

For Chinese character-level constituent parsing

```
conparser
`-- implementations
    `-- jcad
```

Usage

- A tip for training: obtain a best model

```
For i = 1  to  maxN
     ./train    inputfile    modelfile   1
     evaluate on a develop file and get current model's performance
     if(current performance is the best performance)
               save current model
     endif
End for
```

Usage

- More documentation at http://people.sutd.edu.sg/~yue_zhang/doc/index.html

User Manual of ZPar

Yue Zhang
frcchang@gmail.com

March 28, 2013

1 Overview

ZPar is a statistical natural language parser, which performs syntactic analysis tasks including word segmentation, part-of-speech tagging and parsing. ZPar supports multiple languages and multiple grammar formalisms. ZPar has been most heavily developed for Chinese and English, while it provides generic support for other languages. A Romanian model has been trained for ZPar 0.2, for example. ZPar currently supports context free grammars (CFG), dependency grammars and combinatory categorial grammars (CCG).

2 System Requirements

The ZPar software requires the following basic system configuration

- Linux or Mac
- GCC
- 256MB of RAM minimum
- At least 500MB of hard disk space

3 Download and Installation

Download the latest zip files from sourceforge and move them to your work space.
You can use ZPar off the shelf by referring to the quick start, or follow detailed instructions for the compilation, training, and usage of individual modules.

- Chinese word segmentation
- Chinese joint segmentation and POS tagging
- English POS tagging
- Chinese and English dependency parsing
- Chinese and English phrase-structure parsing
- Language- and Treebank-independent parsers
- CCG parsing

4 License

The software source is under GPL (v.3), and a separate commercial license issued by Oxford University for non-opensource. Various models available for download were trained from different text resources, which may require further licenses.

References

[1] Yue Zhang and Stephen Clark. 2011. Syntactic Processing Using the Generalized Perceptron and Beam Search. *Computational Linguistics*, 37(1):105-151.

ZPar

- Introduction
- Usage
- Development
- On-going work
- Contributions welcome

Development

- Add new implementation (dependency parsing as an example)
  - New folder under implementations

```
[mszhang@node06:zpar]$ cd src/common/depparser/implementations/
[mszhang@node06:implementations]$ ls
arceager  covington  eisner  emnlp08  graph_noisy  noisy  punct  uppsala
[mszhang@node06:implementations]$ cp -r arceager newmethod
[mszhang@node06:implementations]$
[mszhang@node06:implementations]$ ls
arceager  covington  eisner  emnlp08  graph_noisy  newmethod  noisy  punct  uppsala
```

Development

- Add new implementation (dependency parsing as an example)
  - New folder under implementations
  - Modify necessary files

```
newmethod
|-- action.h
|-- depparser.cpp
|-- depparser.h
|-- depparser_impl_inc.h
|-- depparser_macros.h
|-- depparser_weight.cpp
|-- depparser_weight.h
`-- state.h
```

Development

- Add new implementation (dependency parsing as an example)
  - New folder under implementations
  - Modify necessary files
  - Modify the Makefile

```
# currently support eisner, covington, nivre, combined and joint implementations
CHINESE_DEPPARSER_IMPL = newmethod
CHINESE_DEPPARSER_LABELED = false
CHINESE_DEPLABELER_IMPL = naive

# currently support sr implementations
CHINESE_CONPARSER_IMPL = jcad

# currently support only agenda
ENGLISH_TAGGER_IMPL = collins

# currently support eisner, covington, nivre, combined implementations
ENGLISH_DEPPARSER_IMPL = newmethod
ENGLISH_DEPPARSER_LABELED = true
ENGLISH_DEPLABELER_IMPL = naive
# currently support sr implementations
ENGLISH_CONPARSER_IMPL = cad
```

Development

- Flexible—give your own Makefile for other tasks

```
-- Makefile
-- Makefile.ccg
-- Makefile.common
-- Makefile.conparser
-- Makefile.deplabeler
-- Makefile.depparser
-- Makefile.doc2snt
-- Makefile.misc
-- Makefile.postagger
-- Makefile.segmentor
-- Makefile.zpar
-- Makefile.zpar.en
-- Makefile.zpar.ge
```

ZPar

- **Introduction**
- **Usage**
- **Development**
- **On-going work**
- **Contributions welcome**

On-going work

- The release of ZPar 0.7 this year
  - New implementations
    - Deep learning POS-tagger (Ma et al., ACL 2014)
    - Character-based Chinese dependency parsing (Zhang et al., ACL 2014)
    - Non-projective parser with more optimizations
    - Double-stack and double-queue models for parsing heterogeneous dependencies (Zhang et al., COLING 2014)

On-going work

- The release of ZPar 0.7 this year
  - New implementations
  - The generic system will replace the Chinese system as the default version

ZPar

- Introduction
- Usage
- Development
- On-going work
- Contributions welcome

Contributions welcome

- Open source contributions
- User interfaces
  - Tokenizer html, ….
- Optimizations
  - Reduced memory usage
  - Parallel versions
  - Microsoft windows versions