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

Yue Zhang\textsuperscript{1}, Meishan Zhang\textsuperscript{2}, Ting Liu\textsuperscript{2}

Singapore University of Technology and Design\textsuperscript{1}
yue\_zhang@sutd.edu.sg
Harbin Institute of Technology, China\textsuperscript{2}
\{mszhang, tliu\}@ir.hit.edu.cn
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Analysis</td>
</tr>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>Analysis</td>
</tr>
</tbody>
</table>
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

```
The guest may come today .
DT    NN    MD    VB    NN
```
Structured prediction problems

- Dependency parsing

ROOT: Economic news had little effect on financial markets.
Structured prediction problems

- Constituent parsing

```
S
  /\      /
 / \    /  \
 NP-SBJ  VP  \
 /  \   /  \
 DT   NN MD  \
 |    |   |  \
 The guest may 
 |      |   |
 VP     NP-TMP 
 |     |   |
 VB come NN  
 |      |   |
 today 
```

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, $p(t_i|t_{i-1}...t_1) = p(t_i|t_{i-1})$
    - 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
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

start
Automata

A transition system

\[ S_1 \]

\[ a_0 \]

start --- arrow --- \( S_1 \)
A transition system

- Automata
Automata

A transition system
A transition system

- Automata
A transition system

- Automata
A transition system

Automata
A transition system

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

- **Actions**
  - The operations that can be applied for state transition
  - Construct output incrementally
    - $a_i$
A transition-based POS-tagging example

- **POS tagging**
  
  I like reading books → I/PRON like/VERB reading/VERB books/NOUN

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

- Start State

I like reading books
A transition-based POS-tagging example

TAG(PRON)

I/PRON  like reading books
A transition-based POS-tagging example

TAG(VERB)

I/PRON like/VERB reading books
A transition-based POS-tagging example

TAG(VERB)

I/PRON like/VERB reading/VERB

books
A transition-based POS-tagging example

- TAG (NOUN)

I/PRON like/VERB reading/VERB books/NOUN
A transition-based POS-tagging example

- End State

```
I/PRON like/VERB reading/VERB books/NOUN
```
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
- Find the best sequence of actions
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: $S_i$
  - Output: $a_i = \arg\max_{a'} w \cdot f(S_i, a')$

For better accuracies: beam-search decoding
Beam-search decoding

start

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
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)
Beam-search decoding

- An example: POS-tagging
  - I like reading books
Beam-search decoding

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

- I like reading books
An example: POS-tagging

- I like reading books

Zhang and Clark, CL 2011
An example: POS-tagging

- I like reading books
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

perceptron update here!

Zhang and Clark, CL 2011
Online learning

**Inputs:** training examples \((x_i, y_i = \{S^i_0, S^i_1, \ldots, S^i_m\})\) is a state sequence \(i=1\ldots N\)

**Initialization:** set \(\overrightarrow{w} = 0\)

**Algorithm:**

```markdown
for \(r = 1 \cdots P, i = 1 \cdots N\) do
    candidates \leftarrow \{S^i_0\}
    agenda \leftarrow \text{CLEAR}(agenda)
    for \(k = 1 \cdots m, m\) corresponds to a specific training example. do
        for each candidate in candidates do
            agenda \leftarrow \text{INSERT}(\text{EXPAND}(candidate), agenda)
            candidates \leftarrow \text{TOP} - B(agenda, B)
            best \leftarrow \text{TOP}(agenda)
            if \(S^i_k\) is not in candidates or (\(best \neq S^i_m\) and \(k\) equals \(m\)) then
                \(\overrightarrow{w} = \overrightarrow{w} + \Phi(S^i_k) - \Phi(best)\)
            end if
        end for
    end for
end for

**Output:** \(\overrightarrow{w}\)
```

Zhang and Clark, CL 2011
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>ZPar</td>
</tr>
</tbody>
</table>
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  
  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)
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

Zhang and Clark, ACL 2007
The transition system

- State
  - Partially segmented results
  - Unprocessed characters

- Two candidate actions
  - Separate  
    
  - Append  

Zhang and Clark, ACL 2007
The transition system

- Initial State

I like reading books

我喜欢读书

Zhang and Clark, ACL 2007
The transition system

- Separate

我

喜欢读书

Zhang and Clark, ACL 2007
The transition system

- Separate

我 喜欢读书
The transition system

- Append

我 喜欢 读书
The transition system

- Separate

书
我喜欢读书
The transition system

- Separate

我 喜欢 读书
The transition system

- End State

```
我 喜欢 读 书
```
Beam search

ABCDE

Candidates

Agenda

Zhang and Clark, ACL 2007
Beam search

BCDE

Candidates

A

Agenda
Beam search

BCDE

A

Candidates

Agenda
Beam search

CDE

Candidates

AB

AB

A

A B

Agenda
Beam search

CDE

Candidates

Agenda
Beam search

CDE

AB
A B

Candidates

Agenda

Zhang and Clark, ACL 2007
Beam search

Candidates

AB
A B

Agenda

ABC
AB C
A BC
A B C
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

<table>
<thead>
<tr>
<th></th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>word $w$</td>
</tr>
<tr>
<td>2</td>
<td>word bigram $w_1w_2$</td>
</tr>
<tr>
<td>3</td>
<td>single character word $w$</td>
</tr>
<tr>
<td>4</td>
<td>a word starting with character $c$ and having length $l$</td>
</tr>
<tr>
<td>5</td>
<td>a word ending with character $c$ and having length $l$</td>
</tr>
<tr>
<td>6</td>
<td>space separated characters $c_1$ and $c_2$</td>
</tr>
<tr>
<td>7</td>
<td>character bigram $c_1c_2$ in any word</td>
</tr>
<tr>
<td>8</td>
<td>the first and last characters $c_1$ and $c_2$ of any word</td>
</tr>
<tr>
<td>9</td>
<td>word $w$ immediately before character $c$</td>
</tr>
<tr>
<td>10</td>
<td>character $c$ immediately before word $w$</td>
</tr>
<tr>
<td>11</td>
<td>the starting characters $c_1$ and $c_2$ of two consecutive words</td>
</tr>
<tr>
<td>12</td>
<td>the ending characters $c_1$ and $c_2$ of two consecutive words</td>
</tr>
<tr>
<td>13</td>
<td>a word with length $l$ and the previous word $w$</td>
</tr>
<tr>
<td>14</td>
<td>a word with length $l$ and the next word $w$</td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th></th>
<th>AS</th>
<th>CU</th>
<th>PU</th>
<th>SAV</th>
<th>OAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01</td>
<td>93.8</td>
<td>90.1</td>
<td>95.1</td>
<td>93.0</td>
<td>95.5</td>
</tr>
<tr>
<td>S04</td>
<td>94.2</td>
<td>89.4</td>
<td>91.8</td>
<td>95.9</td>
<td></td>
</tr>
<tr>
<td>S05</td>
<td>94.5</td>
<td>92.4</td>
<td>92.4</td>
<td>93.1</td>
<td>95.5</td>
</tr>
<tr>
<td>S06</td>
<td>94.5</td>
<td>92.4</td>
<td>93.6</td>
<td>92.9</td>
<td>94.8</td>
</tr>
<tr>
<td>S08</td>
<td>96.1</td>
<td>90.4</td>
<td>94.6</td>
<td>95.4</td>
<td>95.9</td>
</tr>
<tr>
<td>S10</td>
<td>95.9</td>
<td>91.6</td>
<td>94.7</td>
<td>94.7</td>
<td>94/8</td>
</tr>
<tr>
<td>S12</td>
<td>95.6</td>
<td>92.8</td>
<td>94.1</td>
<td>93.8</td>
<td>95.9</td>
</tr>
<tr>
<td>Peng</td>
<td>97.0</td>
<td>94.6</td>
<td>94.6</td>
<td>95.4</td>
<td>95.5</td>
</tr>
<tr>
<td>Z&amp;C 07</td>
<td>97.0</td>
<td>94.6</td>
<td>94.6</td>
<td>95.4</td>
<td>95.5</td>
</tr>
</tbody>
</table>
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**
A dependency structure is a directed graph $G$ with the following constraints:

- Connected
- Acyclic
- Single-head

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

The input
The arc-eager transition system

- Actions
  - Reduce
    - Pops stack

```
... STP

N0 N1 N2 N3 ...

ST

STLC STRC

The stack

The input
```

The stack

ST

STLC STRC

...
The arc-eager transition system

- Actions
  - Arc-Left

![Diagram showing the arc-eager transition system]
The arc-eager transition system

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

```
... STP
```

```
N0 N1 N2 N3 ...
```

```
The stack
```

```
The input
```

```
STLC  STRC
```

```
ST

N0LC
```
The arc-eager transition system

- Actions
  - Arc-right

```
... STP ST STLC STRC N0 N1 N2 N3 ...
```

The stack

The input
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

[Diagram]

He does it here $\rightarrow$ S $\rightarrow$ He does it here
The arc-eager transition system

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

He does it here \(\rightarrow\) S \(\rightarrow\) He \(\rightarrow\) AL \(\rightarrow\) He
The arc-eager transition system

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

He does it here → S → He does it here → AL → He does it here → S → He does it here
An example
- S – Shift
- R – Reduce
- AL – ArcLeft
- AR – ArcRight

The arc-eager transition system
The arc-eager transition system

An example

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

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

The arc-eager transition system
The arc-eager transition system

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

He does it here

He does it here

He does it here

He does it here

He does it here
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

![Diagram of the arc-standard transition system with actions and stack elements.]
The arc-standard transition system

- **Actions**
  - Left-reduce
The arc-standard transition system

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

```
... ST1 STLC STRC N0 N1 N2 N3 ...
```

The stack

The input
The arc-standard transition system

- **Actions**
  - Right-reduce

![Diagram of a transition system with labeled arcs and states such as ST1, ST, STLC, and STRC.]
The arc-standard transition system

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

```
[... ST1 ...]  [N0 N1 N2 N3 ...]
             The stack          The input
            /     \              /     /
           ST    \              STLC   STRC
```

The stack

The input
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

  ![Diagram showing stack and input reordering](image)

  - 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

meeting was scheduled for this today
A transition-based parsing process

- SHIFT

meeting was scheduled for this today
A transition-based parsing process

- **SHIFT**

```
meeting was scheduled for this today
```

A
A transition-based parsing process

- **SHIFT**

  meeting was scheduled for this today
A transition-based parsing process

- SWAP

meeting was for scheduled this today
A transition-based parsing process

- SWAP

A meeting for [A] was scheduled this today
A transition-based parsing process

- **SHIFT**

meeting *for* was scheduled this today
A transition-based parsing process

- **SHIFT**

meeting for was scheduled this today

A
A transition-based parsing process

- **SHIFT**

meeting *for* was scheduled *this* today
A transition-based parsing process

- **SWAP**

- A meeting **for was this** scheduled today
A transition-based parsing process

- SWAP

meeting for this was scheduled today
A transition-based parsing process

- ARC-RIGHT

A meeting for this was scheduled today
A transition-based parsing process

- ARC-RIGHT

A meeting for this was scheduled today
A transition-based parsing process

- **SHIFT**

```
A  meeting was for this
```

scheduled today
A transition-based parsing process

- **ARC-LEFT**

```
was
  `meeting` for
  `A` `this`
```
scheduled today
A transition-based parsing process

- **SHIFT**

```
was scheduled

meeting

A

for

this

today
```
A transition-based parsing process

**SHIFT**

was scheduled today

meeting

A

for

this
A transition-based parsing process

- **ARC-RIGHT**

  was scheduled
  
  meeting  today
  
  A  for
  
  this
A transition-based parsing process

- ARC-RIGHT

```
was
meeting  scheduled
A  for  today
this
```
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

Zhang and Clark, EMNLP 2008
A tale of two parsers

Graph-based

- MST parser
  - Carreras, 2007
  - Comparable
  - Higher order, more features

- Koo and Collins, 2010

Transition-based

- Malt parser
  - Zhang and Clark, 2008
  - This tutorial framework
  - More features

- Zhang and Clark, 2011
  - Comparable
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
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
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

<table>
<thead>
<tr>
<th>from single words</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S0wp; S0w; S0p; N0wp; N0w; N0p; )</td>
</tr>
<tr>
<td>( N1wp; N1w; N1p; N2wp; N2w; N2p; )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from word pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S0wpN0wp; S0wpN0w; S0wN0wp; S0wpN0p; )</td>
</tr>
<tr>
<td>( S0pN0wp; S0wN0w; S0pN0p )</td>
</tr>
<tr>
<td>( N0pN1p )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from three words</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N0pN1pN2p; S0pN0pN1p; S0hpS0pN0p; )</td>
</tr>
<tr>
<td>( S0pS0lpN0p; S0pS0rpN0p; S0pN0pN0lp )</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0wd; S_0pd; N_0wd; N_0pd;$</td>
</tr>
<tr>
<td>$S_0w N_0wd; S_0p N_0pd;$</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th>valency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0wvr; S_0pvr; S_0wvl; S_0pvl; N_0wvl; N_0pvl; $</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th>unigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0hw; S0hp; S0l; S0lw; S0lp; S0ll;</td>
</tr>
<tr>
<td>S0rw; S0rp; S0rl; N0lw; N0lp; N0ll;</td>
</tr>
</tbody>
</table>
The feature templates

The extended features

• Third order
  ➢ Graph-based dependency parsers (Carreras, 2007; Koo and Collins, 2010)

<table>
<thead>
<tr>
<th>third-order</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0h2w; S0h2p; S0h1; S0l2w; S0l2p; S0l2l;</td>
</tr>
<tr>
<td>S0r2w; S0r2p; S0r2l; N0l2w; N0l2p; N0l2l;</td>
</tr>
<tr>
<td>S0pS0lpS0l2p; S0pS0rpS0r2p;</td>
</tr>
<tr>
<td>S0pS0hpS0h2p; N0pN0lpN0l2p;</td>
</tr>
</tbody>
</table>
The feature templates

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

<table>
<thead>
<tr>
<th>label set 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0wsr; S0psr; S0wsl; S0psl;</td>
</tr>
<tr>
<td>N0wsl; N0psl;</td>
</tr>
</tbody>
</table>
Experiments

- **Chinese Data (CTB5)**
  
  Training, development, and test data for Chinese dependency parsing.

<table>
<thead>
<tr>
<th></th>
<th>Sections</th>
<th>Sentences</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>001–815</td>
<td>16,118</td>
<td>437,859</td>
</tr>
<tr>
<td></td>
<td>1,001–1,136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>886–931</td>
<td>804</td>
<td>20,453</td>
</tr>
<tr>
<td></td>
<td>1,148–1,151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>816–885</td>
<td>1,915</td>
<td>50,319</td>
</tr>
<tr>
<td></td>
<td>1,137–1,147</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **English Data (Penn Treebank)**

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

<table>
<thead>
<tr>
<th></th>
<th>Sections</th>
<th>Sentences</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>2–21</td>
<td>39,832</td>
<td>950,028</td>
</tr>
<tr>
<td>Development</td>
<td>22</td>
<td>1,700</td>
<td>40,117</td>
</tr>
<tr>
<td>Test</td>
<td>23</td>
<td>2,416</td>
<td>56,684</td>
</tr>
</tbody>
</table>

Zhang and Clark, ACL 2011
## Results

### Chinese

<table>
<thead>
<tr>
<th>Model</th>
<th>UAS</th>
<th>UEM</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. (2012)</td>
<td>86.8</td>
<td>---</td>
<td>85.4</td>
</tr>
<tr>
<td>Jun et al. (2011)</td>
<td>86.0</td>
<td>35.0</td>
<td>---</td>
</tr>
<tr>
<td>H&amp;S10</td>
<td>85.2</td>
<td>33.7</td>
<td>---</td>
</tr>
<tr>
<td><strong>This Method</strong></td>
<td>86.0</td>
<td>36.9</td>
<td>84.4</td>
</tr>
</tbody>
</table>

### English

<table>
<thead>
<tr>
<th>Model</th>
<th>UAS</th>
<th>UEM</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. (2012)</td>
<td>93.1</td>
<td>---</td>
<td>92.0</td>
</tr>
<tr>
<td>MSTParser</td>
<td>91.5</td>
<td>42.5</td>
<td></td>
</tr>
<tr>
<td>K08 standard</td>
<td>92.0</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>K&amp;C10 model</td>
<td>93.0</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>H&amp;S10</td>
<td>91.4</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>This Method</strong></td>
<td>92.9</td>
<td>48.0</td>
<td>91.8</td>
</tr>
</tbody>
</table>

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
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

<table>
<thead>
<tr>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR布朗</td>
<td>VV访问</td>
</tr>
<tr>
<td>NR上海</td>
<td></td>
</tr>
</tbody>
</table>

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)
The shift-reduce parsing process

- Actions
  - REDUCE-UNARY-X

<table>
<thead>
<tr>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR布朗</td>
<td>VV访问</td>
</tr>
<tr>
<td>NR布朗</td>
<td>NR上海</td>
</tr>
</tbody>
</table>

布朗(Brown) 访问(visits) 上海(Shanghai)

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

- Actions
  - REDUCE-UNARY-X

stack

queue

VV访问       NR上海

NR布朗

布朗(Brown)   访问(visits)   上海(Shanghai)

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

- Actions
  - REDUCE-UNARY-X

stack

queue

布朗(Brown)  访问(visits)  上海(Shanghai)
The shift-reduce parsing process

**Actions**

- REDUCE-BINARY-{$\{L/R\}$}-X

---

stack

queue

NP VV访问 NP

NR布朗 NR上海

布朗(Brown) 访问(visits) 上海(Shanghai)
The shift-reduce parsing process

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

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

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

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

- Actions
  - TERMINATE

---

stack

queue

S

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

- Actions
  - TERMINATE

stack queue ans

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

Example

- SHIFT

stack

queue

NR布朗  VV访问  NR上海
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

<table>
<thead>
<tr>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>VV访问 NR上海</td>
</tr>
<tr>
<td>NR布朗</td>
<td></td>
</tr>
</tbody>
</table>
The shift-reduce parsing process

- Example
  - SHIFT

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

- Example
  - REDUCE-UNARY-NP

```
stack    queue
NP  VV访问  NR上海
NR布朗
```

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

- Example
  - REDUCE-BINARY-L-VP

<table>
<thead>
<tr>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP VV访问 NP</td>
<td>NR布朗 NR上海</td>
</tr>
</tbody>
</table>
The shift-reduce parsing process

- Example
  - REDUCE-BINARY-R-IP

```
stack
 NP  VP
 NR布朗  VV访问  NP
   
 NR上海
```

queue
The shift-reduce parsing process

- Example
  - TERMINATE

Zhang and Clark, IWPT 2009
The shift-reduce parsing process

Example

stack

queue

IP

NP

NR布朗

VV访问

VP

NP

NR上海

Zhang and Clark, IWPT 2009
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

```
Y
/   \
A     B   C   D   E   F
```
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 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 \rightarrow state item 1
The statistical parser

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

Initial item stack: empty
queue: input

SHIFT state item 1

SHIFT state item 2

REDUCE-UNARY-X

state item 3
state item 4

different label

{ state item 3
  state item 4
  ...
  state item N

Zhang and Clark, IWPT 2009
The statistical parser

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

Initial item stack: empty
queue: input

SHIFT state item 1
SHIFT state item 2

REDUCE-UNARY-X

state item 3
state item 4

different label

... state item N

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: $B=1$

Initial item stack: empty queue: input

- SHIFT state item 1
- SHIFT state item 2
- SHIFT

REDUCE-UNARY-X

state item 3
state item 4
... REDUCE-BINARY-{L/R}-X
state item N

different label

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: $B>1$
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.
The statistical parser

- Features
  - Manually combine word and constituent information
    - Unigrams
      $$S_0tc, S_0wc, S_1tc, S_1wc,$$
      $$S_2tc, S_2wc, S_3tc, S_3wc,$$
      $$N_0wt, N_1wt, N_2wt, N_3wt,$$
      $$S_0lwc, S_0rwc, S_0uwc,$$
      $$S_1lwc, S_1rwc, S_1uwc,$$
The statistical parser

- **Features**
  - Manually combine of word and constituent information
    - **Bigrams**
      - $S_0 w S_1 w$, $S_0 w S_1 c$, $S_0 c S_1 w$, $S_0 c S_1 c$,
      - $S_0 w N_0 w$, $S_0 w N_0 t$, $S_0 c N_0 w$, $S_0 c N_0 t$,
      - $N_0 w N_1 w$, $N_0 w N_1 t$, $N_0 t N_1 w$, $N_0 t N_1 t$
      - $S_1 w N_0 w$, $S_1 w N_0 t$, $S_1 c N_0 w$, $S_1 c N_0 t$,
The statistical parser

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

\[ S_0cS_1cS_2c, S_0wS_1cS_2c, S_0cS_1wS_2c, S_0cS_1cS_2w, S_0cS_1cN_0t, S_0wS_1cN_0t, S_0cS_1wN_0t, S_0cS_1cN_0w \]
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
The statistical parser

LEFT: REDUCE-BINARY-R(NP), IDLE
RIGHT: REDUCE-UNARY(NP), REDUCE-BINARY-L(VP)
Experiments

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

- **English results on PTB**

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

Zhu et al., ACL 2013
# Experiments

- **Chinese results on CTB51**

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>LP</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak (2000)</td>
<td>79.6</td>
<td>82.1</td>
<td>80.8</td>
</tr>
<tr>
<td>Bikel (2004)</td>
<td>79.3</td>
<td>82.0</td>
<td>80.6</td>
</tr>
<tr>
<td>Petrov &amp; Klein (2007)</td>
<td>81.9</td>
<td>84.8</td>
<td>83.3</td>
</tr>
<tr>
<td><strong>This implementation</strong></td>
<td><strong>82.1</strong></td>
<td><strong>84.3</strong></td>
<td><strong>83.2</strong></td>
</tr>
</tbody>
</table>
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 ⇒ S
- function composition: (S\NP)/(S\NP) (S\NP)/NP ⇒ (S\NP)/NP

Unary rules change the type of a phrase

- Type raising: NP ⇒ S/(S\NP)
- Type changing: S[pss]\NP ⇒ NP\NP

Zhang and Clark, ACL 2011
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

Zhang and Clark, ACL 2011
Introduction to CCG parsing

- An example derivation

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

S[dcl]\NP

S[dcl]

Zhang and Clark, ACL 2011
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

Zhang and Clark, ACL 2011
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

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**

<table>
<thead>
<tr>
<th>...</th>
<th>$S_2(w_2)$</th>
<th>$S_1(w_1)$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The stack</td>
<td>The queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After **UNARY**

<table>
<thead>
<tr>
<th>...</th>
<th>$S_2(w_2)$</th>
<th>$X(w_1)$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The stack</td>
<td></td>
<td>$S_1(w_1)$</td>
<td>The queue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
```

Zhang and Clark, ACL 2011
The shift-reduce parser

- An example parsing process

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

SHIFT

Zhang and Clark, ACL 2011
The shift-reduce parser

- An example parsing process

```
NP_{IBM}  ((S[dcl]NP)/NP)_{bought}  NP_{Lotus}  yesterday
```

SHIFT

Zhang and Clark, ACL 2011
The shift-reduce parser

- An example parsing process

```
NP_{IBM} (S[dcl]\NP)_{bought}
```

```
(((S[dcl]\NP)/NP)_{bought} NP_{Lotus})
```

yesterday

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

![Parsing Tree]

NP_{IBM} \ (S[dcl] \ NP)_{bought}

(S[dcl] \ NP)_{bought}  (S\NP)\(S\NP)_{yesterday}

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

COMBINE

Zhang and Clark, ACL 2011
The shift-reduce parser

- An example parsing process

```
S[dcl]bought
   NP
   NP
   NP
   NP
   (S[dcl]NP)bought
   (S\NP)(S\NP)_{yesterday}
   (S[dcl]NP)\bought
   ((S[dcl]NP)/NP)bought
   NP_{Lotus}

COMBINE
```

Zhang and Clark, ACL 2011
The shift-reduce parser

- An example parsing process

![Parsing Diagram]

\[\text{S[dcl]}_{bought}\]

\[\text{NP}_{IBM}(\text{S[dcl]}\backslash\text{NP})_{bought}\]

\[\text{(S[dcl]}\backslash\text{NP})_{bought} \quad \text{(S\backslash\text{NP})(S\backslash\text{NP})}_{yesterday}\]

\[\text{(S[dcl]}\backslash\text{NP})_{bought} \quad \text{NP}_{Lotus}\]

FINISH

Zhang and Clark, ACL 2011
**Features**

- **Beam-search decoding**
  - context

  ![Diagram](image)

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

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

Zhang and Clark, ACL 2011
# Test results

- **F&P = Fowler and Penn (2010)**

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

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
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  
  Segment: 我 喜欢 读 书  
  Tag: 我/PN 喜欢/V 读/V 书/N  
  
  I like reading books
  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

<table>
<thead>
<tr>
<th>Input</th>
<th>Segment</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>我喜欢读书</td>
<td>我喜欢读书</td>
<td>我喜欢/N 欢/V 读/V 书/N</td>
</tr>
<tr>
<td>Ilikereadingbooks</td>
<td>Ilke reading books</td>
<td>Il/N ke/V reading/V books/N</td>
</tr>
</tbody>
</table>

- 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 二百三十三/CD (233)
  - 三/CD (3) 百/CD (hundred) 三/CD (3) 十/CD (ten)
Joint segmentation and tagging

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

Input: 我喜欢读书
Output: 我/PN 喜欢/V 读/V 书/N

I 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
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

Zhang and Clark, EMNLP 2010
The transition system

- End state

我/PN 喜欢/V 读/V 书/N
Feature templates for the word segmentor.

<table>
<thead>
<tr>
<th>Feature template</th>
<th>When $c_0$ is</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $w_{-1}$</td>
<td>separated</td>
</tr>
<tr>
<td>2 $w_{-1}w_{-2}$</td>
<td>separated</td>
</tr>
<tr>
<td>3 $w_{-1}$, where $\text{len}(w_{-1}) = 1$</td>
<td>separated</td>
</tr>
<tr>
<td>4 $\text{start}(w_{-1})\text{len}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>5 $\text{end}(w_{-1})\text{len}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>6 $\text{end}(w_{-1})c_0$</td>
<td>separated</td>
</tr>
<tr>
<td>7 $c_{-1}c_0$</td>
<td>appended</td>
</tr>
<tr>
<td>8 $\text{begin}(w_{-1})\text{end}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>9 $w_{-1}c_0$</td>
<td>separated</td>
</tr>
<tr>
<td>10 $\text{end}(w_{-2})w_{-1}$</td>
<td>separated</td>
</tr>
<tr>
<td>11 $\text{start}(w_{-1})c_0$</td>
<td>separated</td>
</tr>
<tr>
<td>12 $\text{end}(w_{-2})\text{end}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>13 $w_{-2}\text{len}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>14 $\text{len}(w_{-2})w_{-1}$</td>
<td>separated</td>
</tr>
</tbody>
</table>

$w =$ word; $c =$ character. The index of the current character is 0.
## Feature templates

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

<table>
<thead>
<tr>
<th>Feature template</th>
<th>when $c_0$ is</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{-1}t_{-1}$</td>
<td>separated</td>
</tr>
<tr>
<td>$t_{-1}t_0$</td>
<td>separated</td>
</tr>
<tr>
<td>$t_{-2}t_{-1}t_0$</td>
<td>separated</td>
</tr>
<tr>
<td>$w_{-1}t_0$</td>
<td>separated</td>
</tr>
<tr>
<td>$t_{-2}w_{-1}$</td>
<td>separated</td>
</tr>
<tr>
<td>$w_{-1}t_{-1}end(w_{-2})$</td>
<td>separated</td>
</tr>
<tr>
<td>$w_{-1}t_{-1}c_0$</td>
<td>separated</td>
</tr>
<tr>
<td>$c_{-2}c_{-1}c_0t_{-1}$, where $\text{len}(w_{-1}) = 1$</td>
<td>separated</td>
</tr>
<tr>
<td>$c_0t_0$</td>
<td>separated</td>
</tr>
<tr>
<td>$t_{-1}\text{start}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>$t_0c_0$</td>
<td>separated or appended</td>
</tr>
<tr>
<td>$c_0t_0\text{start}(w_0)$</td>
<td>appended</td>
</tr>
<tr>
<td>$ct_{-1}\text{end}(w_{-1})$, where $c \in w_{-1}$ and $c \neq \text{end}(w_{-1})$</td>
<td>separated</td>
</tr>
<tr>
<td>$c_0t_0\text{cat}(\text{start}(w_0))$</td>
<td>separated</td>
</tr>
<tr>
<td>$ct_{-1}\text{cat}(\text{end}(w_{-1}))$, where $c \in w_{-1}$ and $c \neq \text{end}(w_{-1})$</td>
<td>appended</td>
</tr>
<tr>
<td>$c_0t_0c_{-1}t_{-1}$</td>
<td>separated</td>
</tr>
<tr>
<td>$c_0t_0c_{-1}$</td>
<td>appended</td>
</tr>
</tbody>
</table>

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

- Penn Chinese Treebank 5 (CTB-5)

<table>
<thead>
<tr>
<th></th>
<th>CTB files</th>
<th># sent.</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1-270</td>
<td>18089</td>
<td>493,939</td>
</tr>
<tr>
<td></td>
<td>400-1151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop</td>
<td>301-325</td>
<td>350</td>
<td>6,821</td>
</tr>
<tr>
<td>Test</td>
<td>271-300</td>
<td>348</td>
<td>8,008</td>
</tr>
</tbody>
</table>
Experiments

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

<table>
<thead>
<tr>
<th></th>
<th>SF</th>
<th>JF</th>
</tr>
</thead>
<tbody>
<tr>
<td>K09 (error-driven)</td>
<td>97.87</td>
<td>93.67</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td><strong>97.78</strong></td>
<td><strong>93.67</strong></td>
</tr>
<tr>
<td>Zhang 2008</td>
<td>97.82</td>
<td>93.62</td>
</tr>
<tr>
<td>K09 (baseline)</td>
<td>97.79</td>
<td>93.60</td>
</tr>
<tr>
<td>J08a</td>
<td>97.85</td>
<td>93.41</td>
</tr>
<tr>
<td>J08b</td>
<td>97.74</td>
<td>93.37</td>
</tr>
<tr>
<td>N07</td>
<td>97.83</td>
<td>93.32</td>
</tr>
</tbody>
</table>

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

Zhang and Clark, EMNLP 2010
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:

    He/PN → does/V → it/PN → here/RB

- Accurate dependency parsing heavily relies on POS tagging information

- Error propagation

- Syntactic information can be helpful for POS disambiguation
Joint POS-tagging and dependency parsing

- Input: POS-tagged sentence  
  e.g. He does it here
- Output:

He/PN  does/V  it/PN  here/RB
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)

---

Hatori et al. IJCNLP 2011
The extended arc-standard transition system

- Actions
  - SHIFT(t)
    - Pushes stack

```plaintext
The stack

ST1
STLC
ST
STRC
N0/t
N1 N2 N3 ...

The input
```

Hatori et al. IJCNLP 2011
The extended arc-standard transition system

- **Actions**
  - LEFT-REDUCE

![Diagram of the extended arc-standard transition system with actions](image)

*Hatori et al. IJCNLP 2011*
The extended arc-standard transition system

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

The stack

The input

Hatori et al. IJCNLP 2011
The extended arc-standard transition system

- Actions
  - RIGHT-REDUCE
The extended arc-standard transition system

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

![Diagram of arc-standard transition system]

The extended arc-standard transition system is a formal model used in computational linguistics, particularly for natural language processing tasks. It extends the basic arc-standard transition system with additional actions and rules to handle more complex linguistic structures. The diagram illustrates the interactions between the stack (`ST`), the stack contents (`STLC` and `STRC`), and the input stream (`N0 N1 N2 N3 ...`).

Hatori et al. IJCNLP 2011
The extended arc-standard transition system

- An example
  - $S(t) \rightarrow \text{SHIFT}(t)$
  - LR $\rightarrow$ LEFT-REDUCE
  - RR $\rightarrow$ RIGHT-REDUCE

He does it here
The extended arc-standard transition system

**An example**

- $S(t) \rightarrow \text{SHIFT}(t)$
- $LR \rightarrow \text{LEFT-REDUCE}$
- $RR \rightarrow \text{RIGHT-REDUCE}$

He does it here $S(PN) \rightarrow$ \underline{He/PN} does it here
The extended arc-standard transition system

- An example
  - $S(t) \rightarrow \text{SHIFT}(t)$
  - LR $\rightarrow \text{LEFT-REDUCE}$
  - RR $\rightarrow \text{RIGHT-REDUCE}$

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 → LR

Does/V

He/PN

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
S(V)  He/PN does/V
LR  Does/V
He/PN

He/PN  does/V
here
He/PN  it/PN
RR  does/V it/PN
He/PN

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

The extended arc-standard transition system

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

The extended arc-standard transition system
The extended arc-standard transition system

- An example
  - $S(t) \rightarrow$ SHIFT(t)
  - LR \rightarrow LEFT-REDUCE
  - RR \rightarrow RIGHT-REDUCE

Hatori et al. IJCNLP 2011
Features

POS tag features

\[
\begin{align*}
& t \circ w_j & t \circ t_{j-1} \\
& t \circ t_{j-1} \circ t_{j-2} & t \circ w_j \circ w_j + 1 \\
& t \circ w_j \circ E(w_{j-1}) & t \circ w_j \circ B(w_{j+1}) \\
& t \circ E(w_{j-1}) \circ w_j \circ B(w_{j+1}) & \text{(if len}(w_j) = 1) \\
& t \circ B(w_j) & t \circ E(w_j) \\
& t \circ C_n(w_j) & (n \in \{2, \ldots, \text{len}(w_j) - 1\}) \\
& t \circ B(w_j) \circ C_n(w_j) & (n \in \{2, \ldots, \text{len}(w_j)\}) \\
& t \circ E(w_j) \circ C_n(w_j) & (n \in \{1, \ldots, \text{len}(w_j) - 1\}) \\
& t \circ C_n(w_j) & \text{(if } C_n(w_j) \text{ equals to } C_{n+1}(w_j)) \\
& t \otimes P(B(w_j)) & t \otimes P(E(w_j)) \\
\end{align*}
\]
Features

Dependency parsing features

| (a)  | $s_0.w$ | $s_0.t$ | $s_0.w \odot s_0.t$ | $s_1.w$ | $s_1.t$ | $s_1.w \odot s_1.t$ | $q_0.w$ | $q_0.t$ | $q_0.w \odot q_0.t$ | $s_0.t \odot s_1.t$ | $s_0.w \odot s_0.t \odot s_1.t$ | $s_0.w \odot s_0.t \odot s_1.w \odot s_1.t$ | $s_0.w \odot q_0.t \odot q_1.t$ | $s_1.t \odot s_0.t \odot q_0.t$ | $s_1.t \odot s_0.w \odot q_0.t$ | $s_1.t \odot s_1.lc.t \odot s_0.t$ | $s_1.t \odot s_1.lc.t \odot s_0.w$ | $s_1.t \odot s_0.t \odot s_0.rc.t$ | $s_2.t \odot s_1.t \odot s_0.t$ |
|------|---------|---------|---------------------|---------|---------|---------------------|---------|---------|---------------------|-------------------|------------------|------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| (b)  | $s_0.w \odot d$ | $s_0.t \odot d$ | $s_1.w \odot d$ | $s_0.w \odot s_0.v_l$ | $s_1.w \odot s_1.v_r$ | $s_1.t \odot s_1.v_l$ | $s_1.w \odot s_1.v_l$ | $s_0.lc.w$ | $s_0.lc.t$ | $s_1.rc.w$ | $s_1.rc.t$ | $s_0.lc_2.w$ | $s_0.lc_2.t$ | $s_1.rc_2.w$ | $s_1.rc_2.t$ | $s_0.t \odot s_0.lc.t \odot s_0.lc_2.t$ | $s_1.t \odot s_1.rc.t \odot s_1.rc_2.t$ | $s_1.t \odot s_1.lc.t \odot s_1.lc_2.t$ | $s_1.t \odot s_1.lc.t \odot s_1.lc_2.t$ |

Hatori et al. IJCNLP 2011
Features

Syntactic features

\[ t \circ s_0.w \circ s_0.t \circ s_0.rc.t \]
\[ t \circ s_0.w \circ q_0.w \circ s_0.t \circ s_0.rc.t \]
\[ t \circ B(s_0.w) \circ q_0.w \circ s_0.t \circ s_0.rc.t \]
\[ t \circ s_0.t \circ s_0.rc.t \]
\[ t \circ E(s_0.w) \circ q_0.w \circ s_0.t \circ s_0.lc.t \]
\[ t \circ s_0.t \circ s_0.lc.t \]
## Experiments

- **CTB5 dataset**

<table>
<thead>
<tr>
<th></th>
<th>Sections</th>
<th>Sentences</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>001–815</td>
<td>16,118</td>
<td>437,859</td>
</tr>
<tr>
<td></td>
<td>1,001–1,136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>886–931</td>
<td>804</td>
<td>20,453</td>
</tr>
<tr>
<td></td>
<td>1,148–1,151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>816–885</td>
<td>1,915</td>
<td>50,319</td>
</tr>
<tr>
<td></td>
<td>1,137–1,147</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>LAS</th>
<th>UAS</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. (2011) (unlabeled)</td>
<td></td>
<td>80.74</td>
<td>93.08</td>
</tr>
<tr>
<td>Li et al. (2012) (unlabeled)</td>
<td>---</td>
<td>81.21</td>
<td>94.51</td>
</tr>
<tr>
<td>Li et al. (2012) (labeled)</td>
<td>79.01</td>
<td>81.67</td>
<td>94.60</td>
</tr>
<tr>
<td>Hatori et al. (2011) (unlabeled)</td>
<td>---</td>
<td>81.33</td>
<td>93.94</td>
</tr>
<tr>
<td>Bohnet and Nirve (2012) (labeled)</td>
<td>77.91</td>
<td>81.42</td>
<td>93.24</td>
</tr>
<tr>
<td><strong>Our implementation (unlabeled)</strong></td>
<td>---</td>
<td>81.20</td>
<td>94.15</td>
</tr>
<tr>
<td><strong>Our implementation (labeled)</strong></td>
<td>78.30</td>
<td>81.26</td>
<td>94.28</td>
</tr>
</tbody>
</table>
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 hierarchical 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.
Why character-based?

- Deep character information of word structures.
Why character-based?

- Deep character information of word structures.

Representing the whole word by a character, which is less sparse.

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.
Word structure annotation

- Binarized tree structure for each word.

```
NN-l
  NN-c
    NN-b
      朋
      (friend)
  NN-i
    友
    (friend)
  们
  (plural)

NN-r
  NN-c
    NN-b
      教
      (teach)
  NN-i
    育
    (education)
  界
  (field)
```
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.
Word structure annotation

- Binarized tree structure for each word.

\[ \text{NN-}b, \text{NN-}i, \text{NN-}l, \text{NN-}r, \text{NN-}c \text{ denote the head direction of current node, respectively left, right and coordination.} \]

\[ \text{b, i denote whether the below character is at a word’s beginning position.} \]

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.

```
NN-r
  NN-b
  |    |
  |    |
制 服
```

```
NN-i
  NN-b
  |    |
  |    |
制 服
```

```
VV-l
  VV-b
  |    |
  |    |
制 服
```

```
VV-i
  VV-b
  |    |
  |    |
制 服
```

uniform dress  dominate

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.
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.
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.
The transition system

- **State:**
  - Stack: \( S_0, S_{1l}, S_{1r}, S_{2} \)
  - Queue: \( Q_0, Q_{1} \)

- **Actions:**
  - \( \text{SHIFT-SEPARATE}(t) \)
  - \( \text{SHIFT-APPEND} \)
  - \( \text{REDUCE-SUBWORD}(d) \)
  - \( \text{REDUCE-WORD} \)
  - \( \text{REDUCE-BINARY}(d;l) \)
  - \( \text{REDUCE-UNARY}(l) \)
  - \( \text{TERMINATE} \)
**Actions**

- **SHIFT-SEPARATE(t)**

Diagram:

```
stack
... NP
| NR-t
| NR-r
| NR-b NR-i
| 中 国
```

```
queue
... 银行 (construction) (building)
```

Zhang et al. ACL 2013
SHIFT-SEPARATE(t)
Actions

- **SHIFT-APPEND**
 Actions

- **SHIFT-APPEND**

```
... NP  NN-b  stack  queue  stack  queue ...
  NR-t     建 (building)    业 (industry)    建 (construction)  筑 (building) ...
   NR-r     (construction)   ...    ...     (industry) ...
      NR-b  NR-i      中 (middle)  国 (nation)      中 (middle)  国 (nation)
```

Zhang et al. ACL 2013
**Actions**

- **REDUCE-SUBWORD(d)**

![Diagram showing stack and queue with Chinese and English words]
**Actions**

- **REDUCE-SUBWORD(d)**

  ![Diagram showing the process of reducing subwords](image)

Zhang et al. ACL 2013
Actions

- **REDUCE-WORD**

Zhang et al. ACL 2013
**Actions**

**REDUCE-WORD**

```
... NP
   NR-t
   NR-r
   NR-b NR-i
       建筑 (construction) (building)

stack

queue 

... 呈 (present) ...  

... NP
   NR-t
   NR-r
   NR-b NR-i
       建筑 (construction) (building)

stack

queue 

Zhang et al. ACL 2013
```
Actions

- REDUCE-BINARY(d; l)
Actions

- **REDUCE-BINARY**(d; l)

Zhang et al. ACL 2013
REDUCE-UNARY(1)
Actions

- REDUCE-UNARY(1)

Zhang et al. ACL 2013
- **TERMINATE**
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
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**
## Experiments

- Penn Chinese Treebank 5 (CTB-5)

<table>
<thead>
<tr>
<th></th>
<th>CTB files</th>
<th># sent.</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1-270</td>
<td>18089</td>
<td>493,939</td>
</tr>
<tr>
<td></td>
<td>400-1151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop</td>
<td>301-325</td>
<td>350</td>
<td>6,821</td>
</tr>
<tr>
<td>Test</td>
<td>271-300</td>
<td>348</td>
<td>8,008</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Task</th>
<th>Seg</th>
<th>Tag</th>
<th>Parse</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pipeline</strong></td>
<td>97.35</td>
<td>93.51</td>
<td>81.58</td>
</tr>
<tr>
<td></td>
<td>98.02</td>
<td>94.15</td>
<td>82.95</td>
</tr>
<tr>
<td></td>
<td>97.69</td>
<td>93.83</td>
<td>82.26</td>
</tr>
<tr>
<td><strong>Flat word structures</strong></td>
<td>97.32</td>
<td>94.09</td>
<td>83.39</td>
</tr>
<tr>
<td></td>
<td>98.13</td>
<td>94.88</td>
<td>83.84</td>
</tr>
<tr>
<td></td>
<td>97.73</td>
<td>94.48</td>
<td>83.61</td>
</tr>
<tr>
<td><strong>Annotated word structures</strong></td>
<td>97.49</td>
<td>94.46</td>
<td>84.42</td>
</tr>
<tr>
<td></td>
<td>98.18</td>
<td>95.14</td>
<td>84.43</td>
</tr>
<tr>
<td></td>
<td>97.84</td>
<td>94.80</td>
<td>84.43</td>
</tr>
<tr>
<td><strong>WS</strong></td>
<td>94.02</td>
<td>94.69</td>
<td>94.35</td>
</tr>
</tbody>
</table>

Zhang et al. ACL 2013
## Compare with other systems

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

Diagram:

- 林业局 (forestry administration)
- 副局长 (deputy director)
- 会 (meeting)
- 上 (in)
- 发言 (make a speech)
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
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

<table>
<thead>
<tr>
<th>step</th>
<th>action</th>
<th>stack</th>
<th>queue</th>
<th>dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>φ</td>
<td></td>
<td>φ</td>
</tr>
<tr>
<td>1</td>
<td>SH_w(NR)</td>
<td>林/NR</td>
<td>业 局 ...</td>
<td>A_1 = {林⇒业}</td>
</tr>
<tr>
<td>2</td>
<td>SH_c</td>
<td>林/NR  业/NR</td>
<td>局 副 ...</td>
<td>A_1</td>
</tr>
<tr>
<td>3</td>
<td>AL_c</td>
<td>业/NR</td>
<td></td>
<td>A_2 = A_1 ∪{业⇒局}</td>
</tr>
<tr>
<td>4</td>
<td>SH_c</td>
<td>业/NR  局/NR</td>
<td>副 局 ...</td>
<td>A_2</td>
</tr>
<tr>
<td>5</td>
<td>AL_c</td>
<td>局/NR</td>
<td></td>
<td>A_2</td>
</tr>
<tr>
<td>6</td>
<td>PW</td>
<td>林业局/NR</td>
<td>副 局 ...</td>
<td>A_2</td>
</tr>
<tr>
<td>7</td>
<td>SH_w(NN)</td>
<td>林业局/NR 副/NN</td>
<td>局 长 ...</td>
<td>A_2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>12</td>
<td>PW</td>
<td>林业局/NR 副局长/NN</td>
<td>会上 ...</td>
<td>A_i</td>
</tr>
<tr>
<td>13</td>
<td>AL_w</td>
<td>副局长/NN</td>
<td>会上 ...</td>
<td>A_{i+1} = A_i ∪{林业局/NR⇒副局长/NN}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Main method

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

Zhang et al. ACL 2014
Word-level to character-level

- Arc-eager

<table>
<thead>
<tr>
<th>step</th>
<th>action</th>
<th>stack</th>
<th>deque</th>
<th>queue</th>
<th>dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>φ</td>
<td>林业 …</td>
<td>林业 …</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SHc(NR)</td>
<td>φ</td>
<td>林/NR</td>
<td>业 局 …</td>
<td>φ</td>
</tr>
<tr>
<td>2</td>
<td>ALc</td>
<td>φ</td>
<td>φ</td>
<td>业/NR 局 …</td>
<td>A1 = {业^业}</td>
</tr>
<tr>
<td>3</td>
<td>SHc</td>
<td>φ</td>
<td>业/NR</td>
<td>局 副 …</td>
<td>A1</td>
</tr>
<tr>
<td>4</td>
<td>ALc</td>
<td>φ</td>
<td>φ</td>
<td>局/NR 副 …</td>
<td>A2 = A1 ∪{业^局}</td>
</tr>
<tr>
<td>5</td>
<td>SHc</td>
<td>φ</td>
<td>局/NR</td>
<td>副 局 …</td>
<td>A2</td>
</tr>
<tr>
<td>6</td>
<td>PW</td>
<td>φ</td>
<td>林业局/NR</td>
<td>副 局 …</td>
<td>A2</td>
</tr>
<tr>
<td>7</td>
<td>SHw</td>
<td>林业局/NR</td>
<td>φ</td>
<td>副 局 …</td>
<td>A2</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>13</td>
<td>PW</td>
<td>林业局/NR</td>
<td>副局长/NN</td>
<td>会上 …</td>
<td>Ai</td>
</tr>
<tr>
<td>14</td>
<td>ALw</td>
<td>φ</td>
<td>副局长/NN</td>
<td>会上 …</td>
<td>Ai+1 = Ai ∪{林业局/NR^副局长/NN}</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Main method

- New features

<table>
<thead>
<tr>
<th>Feature templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_c$, $L_{ct}$, $R_c$, $R_{ct}$, $L_{lc1c}$, $L_{rc1c}$, $R_{lc1c}$, $L_c \cdot R_c$, $L_{lc1ct}$, $L_{rc1ct}$, $R_{lc1ct}$, $L_c \cdot R_w$, $L_w \cdot R_c$, $L_{ct} \cdot R_w$, $L_{wt} \cdot R_c$, $L_w \cdot R_{ct}$, $L_c \cdot R_{wt}$, $L_c \cdot R_c \cdot L_{lc1c}$, $L_c \cdot R_c \cdot L_{rc1c}$, $L_c \cdot R_c \cdot L_{lc2c}$, $L_c \cdot R_c \cdot L_{rc2c}$, $L_c \cdot R_c \cdot R_{lc1c}$, $L_c \cdot R_c \cdot R_{lc2c}$, $L_{lsw}$, $L_{rsw}$, $R_{lsw}$, $R_{rsw}$, $L_{lswt}$, $L_{rswt}$, $R_{lswt}$, $R_{rswt}$, $L_{lsw} \cdot R_w$, $L_{rsw} \cdot R_w$, $L_w \cdot R_{lsw}$, $L_w \cdot R_{rsw}$</td>
</tr>
</tbody>
</table>

Zhang et al. ACL 2014
Experiments

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

<table>
<thead>
<tr>
<th></th>
<th>CTB50</th>
<th>CTB60</th>
<th>CTB70</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#sent</td>
<td>18k</td>
<td>23k</td>
<td>31k</td>
</tr>
<tr>
<td>#word</td>
<td>494k</td>
<td>641k</td>
<td>718k</td>
</tr>
<tr>
<td><strong>Development</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#sent</td>
<td>350</td>
<td>2.1k</td>
<td>10k</td>
</tr>
<tr>
<td>#word</td>
<td>6.8k</td>
<td>60k</td>
<td>237k</td>
</tr>
<tr>
<td>#oov</td>
<td>553</td>
<td>3.3k</td>
<td>13k</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#sent</td>
<td>348</td>
<td>2.8k</td>
<td>10k</td>
</tr>
<tr>
<td>#word</td>
<td>8.0k</td>
<td>82k</td>
<td>245k</td>
</tr>
<tr>
<td>#oov</td>
<td>278</td>
<td>4.6k</td>
<td>13k</td>
</tr>
</tbody>
</table>

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

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

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
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>ZPar</td>
</tr>
</tbody>
</table>
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

Zhang and Nivre, COLING 2012
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
      - Classifier

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

Zhang and Nirve, COLING 2012
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 $\approx$ local learning
  - Beam $> 1$, global learning $\uparrow$, local learning $\downarrow$
  - Richer features, make $\uparrow$ or $\downarrow$ faster.

Zhang and Nivre, COLING 2012
Empirical analysis

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

<table>
<thead>
<tr>
<th>training beam</th>
<th>testing beam</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>89.04</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>79.34</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>87.07</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>92.27</td>
</tr>
</tbody>
</table>
Empirical analysis

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

<table>
<thead>
<tr>
<th></th>
<th>ZPar</th>
<th>Malt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>92.18</td>
<td>89.37</td>
</tr>
<tr>
<td>+distance</td>
<td>+0.07</td>
<td>-0.14</td>
</tr>
<tr>
<td>+valency</td>
<td>+0.24</td>
<td>0.00</td>
</tr>
<tr>
<td>+unigrams</td>
<td>+0.40</td>
<td>-0.29</td>
</tr>
<tr>
<td>+third-order</td>
<td>+0.18</td>
<td>0.00</td>
</tr>
<tr>
<td>+label set</td>
<td>+0.07</td>
<td>+0.06</td>
</tr>
<tr>
<td>Extended</td>
<td>93.14</td>
<td>89.00</td>
</tr>
</tbody>
</table>
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

Zhang and Nivre, COLING 2012
Analysis

- Empirical analysis
- Theoretical analysis
Theoretical analysis

- The perceptron
  - Online learning framework

**Inputs:** training examples \( (x_i, y_i) |_{i=1}^{T} \)

**Initialization:** set \( \vec{w} \)

**Algorithm:**

for \( r = 1 \cdots \)

for \( i = 1 \cdots \)

- calculate \( z_i = \text{decode}(w, x_i) \)
- if \( z_i \neq y_i \)
  - \( \vec{w} \leftarrow \vec{w} + \lambda \phi(x_i, z_i) \)

**output:** \( \vec{w} \)
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 \(R^2/\lambda^2\)

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

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

if \(u\) can separate 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 \((x_t, y_t)\) 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 \(R^2 / \lambda^2\)

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

if \(u\) can separate the data, then

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

the margin

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

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

then \(w^{k+1} \geq k\lambda\)
Theoretical analysis

- The perceptron

  - If the data \((x_t, y_t)\) 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 \(R^2/\lambda^2\)

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

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

  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 \(\|w^{k+1}\|^2 \leq 4kR^2\)
Theoretical analysis

The perceptron

- If the data \((x_t, y_t)\) for all \(t = 1, 2, 2, 2\) 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 \(R^2 / \lambda^2\)

\[
\begin{align*}
w^{k+1} &= w^k + (\phi(x_t, y_t) - \phi(x_t, y^p)) \\
\|w^{k+1}\|^2 &= \|w^k\|^2 + 2(\phi(x_t, y_t) - \phi(x_t, y^p))w^k + \|\phi(x_t, y_t) - \phi(x_t, y^p)\|^2
\end{align*}
\]

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

This is satisfied in dynamic programming, it may not hold in beam-search
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 $R^2/\lambda^2$

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

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

$$k \leq \frac{4R^2}{\lambda^2}$$, another words, also $k \leq \frac{R^2}{\lambda^2}$
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 \(o(T)\).

\[
\begin{align*}
w^{k+1}u &= (w^k + (\phi(x_t, y_t) - \phi(x_t, y^p)))u \\
&= w^k u + (\phi(x_t, y_t) - \phi(x_t, y^p))u
\end{align*}
\]

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

1: repeat
2: for each example \((x, y)\) in \(D\) do
3: \(z \leftarrow \text{EXACT}(x, w)\)
4: if \(z \neq y\) then
5: \(w \leftarrow w + \Delta \Phi(x, y, z)\)
6: until converged

perceptron update:
\[
\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta \Phi(x, y, z)
\]
\[
\mathbf{u} \cdot \mathbf{w}^{(k+1)} = \mathbf{u} \cdot \mathbf{w}^{(k)} + \mathbf{u} \cdot \Delta \Phi(x, y, z) \\
\geq \delta \quad \text{margin}
\]

(by induction)
\[
||\mathbf{u}|| \cdot ||\mathbf{w}^{(k+1)}|| \geq \mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta
\]
\[
||\mathbf{w}^{(k+1)}|| \geq k\delta
\]

Huang et al., NAACL 2012
Theoretical analysis

The perceptron

1: repeat
2: for each example \((x, y)\) in \(D\) do
3: \(z \leftarrow \text{EXACT}(x, w)\)
4: if \(z \neq y\) then
5: \(w \leftarrow w + \Delta \Phi(x, y, z)\)
6: until converged

Violation: incorrect label scored higher

perceptron update:
\[
\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta \Phi(x, y, z)
\]

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

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

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

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

\[
\|w^{(k)}\|^2 + \|\Delta \Phi(x, y, z)\|^2 \leq R^2 \\
\text{diameter}
\leq 0 \\
\text{violation}
\]
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 (pruned)
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>ZPar</td>
</tr>
</tbody>
</table>
ZPar

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

- Brief introduction
- Usage
- Development
- On-going work
- Contributions welcome
Initiated in 2009 at Oxford, extended at Cambridge and SUTD, with more developers being involved

ZPar
Brought to you by: freecang

Description

ZPar statistical parser. Universal language support (depending on the availability of training data), with language-specific features for Chinese and English. Currently support word segmentation, POS tagging, dependency and phrase-structure parsing.
Brief introduction

- 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

- Functionalities extended
- Released several versions

<table>
<thead>
<tr>
<th>Name</th>
<th>Modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>2013-09-17</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>2011-11-18</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>2010-09-27</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>2010-04-16</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>2010-03-23</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>2009-09-28</td>
<td></td>
</tr>
</tbody>
</table>
Brief introduction

- 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

- Functionalities extended
- Released several versions
- Contains all implementations of this tutorial
- Code structure
ZPar

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

Download

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Modified</th>
<th>Size</th>
<th>Downloads / Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>zpar.zip</td>
<td>2013-09-17</td>
<td>3.7 MB</td>
<td>8</td>
</tr>
<tr>
<td>english.zip</td>
<td>2013-09-04</td>
<td>189.1 MB</td>
<td>4</td>
</tr>
<tr>
<td>chinese.zip</td>
<td>2013-09-04</td>
<td>575.0 MB</td>
<td>1</td>
</tr>
</tbody>
</table>

Totals: 3 Items 767.8 MB 13
Usage

For off-the-shelf Chinese language processing:

- Compile: make zpar

```bash
[mzs@node06:zpar]$ make zpar
mkdir -p ./obj
mkdir -p ./dist

g++ -W -03 -I./src/include -DDEBUG -I./src/chinese -c ./src/chinese/doc2snt/doc2snt.cpp -o ./obj/chinese/doc2snt.obj ./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 -03 -I./src/include -DDEBUG -c src/libs/reader.cpp -o obj/reader.obj
mkdir -p ./obj
mkdir -p ./obj/linguistics

g++ -W -03 -I./src/include -DDEBUG -c src/libs/writer.cpp -o obj/writer.obj

./src/include/linguistics/cfgtemp.h:26: warning: base class 'class chinese::CConstituentLabel' should be in the copy constructor

compile: make zpar

The Chinese zpar system compiled successfully into ./dist.
```
For off-the-shelf Chinese language processing:

- Compile: make zpar
- Usage

```bash
$ cd dist/
$ ls
$ ./.zpar
```

Usage: `.zpar feature_path [input_file [outout_file]]`
Options:
- `-o[s|t|d]|d|c` : output format, ’s’ segmented format, ’t’ pos-tagged format in sentences, ’td’ pos-tagged format in documents with std::cout sentence boundary delimation, ’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

<table>
<thead>
<tr>
<th>Name</th>
<th>Modified</th>
<th>Size</th>
<th>Downloads / Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>zpar.zip</td>
<td>2013-09-17</td>
<td>3.7 MB</td>
<td>8</td>
</tr>
<tr>
<td>english.zip</td>
<td>2013-09-04</td>
<td>189.1 MB</td>
<td>1</td>
</tr>
<tr>
<td>chinese.zip</td>
<td>2013-09-04</td>
<td>575.0 MB</td>
<td>1</td>
</tr>
</tbody>
</table>
Usage

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

```
mszhang@node01:~$ cd /zpar ..;/chinese
mszhang@node01:~/zpar ..;/chinese$ make
mszhang@node01:~/zpar ..;/chinese$ /zpar ..;/chinese
Initializing ZPar...
[The segmentation and tagging model] Loading scores ... set character knowledge... done. (10.94s)
[The parsing model] Loading scores... done. (10.06s)
ZPar initialized.
```

```
mszhang@node01:~/zpar ..;/chinese$ ./zpar ..;/chinese -cd
mszhang@node01:~/zpar ..;/chinese$ ./zpar ..;/chinese
```

```
这是一个例子。
```

```
这是一个例子。
```

```
more
```

```
more
```

```
more
```
Usage

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

```bash
$ make zpar.en
mkdir -p ./obj
mkdir -p ./obj/linguistics
g++ -W -03 -I./include -DNDEBUG -c src/libs/reader.cpp -o obj/reader.o
mkdir -p ./obj
mkdir -p ./obj/linguistics
g++ -W -03 -I./include -DNDEBUG -c src/libs/writer.cpp -o obj/writer.o
src/libs/writer.cpp: In member function ‘void CSentenceWriter::writeSentence(const CStringVector*, const std::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 constructor
g++ -o ./dist/zpar.en ./obj/zpar.en.o ./obj/english.postagger/weight.o ./obj/english.postagger.o ./obj/english.depparser.o ./obj/english.depparser/weight.o ./obj/english.depparser.o ./obj/english.conparser/constituent.o ./obj/english.conparser/weight.o ./obj/english.deplabeler.o ./obj/english.deplabeler/weight.o ./obj/english.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

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

Usage: ./zpar.en feature_path [input_file [output_file]]

Options:
- `-o[t|d|c]`: output 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
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))) (. .))
```
A generic ZPar

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

- Compile: make zpar.ge
- Usage

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]`: output format; ’t’ pos-tagged format in sentences, ’d’ refers to labeled dependency tree format, and ’c’ refers to constituent parse tree format. Default: d;
For generic processing:

- Compile: make zpar.ge
- Usage
- An example

```
[mszhang@node06:dist]$ tree ../english
  ../english
    |-- comparser
    |    |-- 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
      ```
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

For Chinese character-level constituent parsing
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

User Manual of ZPar

Yue Zhang
frzchung@gmail.com
March 25, 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 (CCCG).

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 word segmentation and POS tagging
- English POS tagging
- Chinese and English dependency parsing
- Chinese and English phrase-structure parsing
- Language- and Treebank-independent parsing
- CRF parsing

4 License

The software source is under GPL (v3), and a separate commercial license is issued by Oxford University for non-open-source. Various models available for download were trained from different text resources, which may require further licenses.

References

ZPar

- Introduction
- Usage
- Development
- On-going work
- Contributions welcome
Add new implementation (dependency parsing as an example)

- New folder under implementations

```
$ cd src/common/depparser/implementations/
$ ls
arceager covington eisner emnlp08 graph_noisy noisy punct uppsala
$ cp -r arceager newmethod
$ ls
arceager covington eisner emnlp08 graph_noisy newmethod noisy punct uppsala
```
Add new implementation (dependency parsing as an example)

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

- New folder under implementations
- Modify necessary files
- Modify the Makefile

```bash
# 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
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