Marrying Dynamic Programming with Recurrent Neural Networks

I eat sushi with tuna from Japan

Liang Huang
Oregon State University
Structured Prediction Workshop, EMNLP 2017, Copenhagen, Denmark
Marrying Dynamic Programming with Recurrent Neural Networks

I eat sushi with tuna from Japan

Liang Huang
Oregon State University
Structured Prediction Workshop, EMNLP 2017, Copenhagen, Denmark
Marrying Dynamic Programming with Recurrent Neural Networks

I eat sushi with tuna from Japan

Liang Huang
Oregon State University

Structured Prediction Workshop, EMNLP 2017, Copenhagen, Denmark
Structured Prediction is Hard!
Not Easy for Humans Either...

This is not a toy and should be kept away from children made in China

(structural ambiguity :-P)
Not Even Easy for Nature!

- prion: “misfolded protein”
- structural ambiguity for the same amino-acid sequence
- similar to different interpretations under different contexts
- causes mad-cow diseases etc.

© Mayo Foundation for Medical Education and Research. All rights reserved.
Case Study: Parsing and Folding

• both problems have exponentially large search space
• both can be modeled by grammars (context-free & above)
• question 1: how to search for the highest-scoring structure?
• question 2: how to make gold structure score the highest?

I eat sushi with tuna from Japan

I eat sushi with tuna from Japan
Solutions to Search and Learning

• question 1: how to search for the highest-scoring structure?
  • answer: dynamic programming to factor search space

• question 2: how to make gold structure score the highest?
  • answer: neural nets to automate feature engineering

• But do DP and neural nets like each other??

I eat sushi with tuna from Japan
Solutions to Search and Learning

• question 1: how to search for the highest-scoring structure?
  • answer: **dynamic programming to factor search space**

• question 2: how to make gold structure score the highest?
  • answer: **neural nets to automate feature engineering**

• But do DP and neural nets like each other??

I eat sushi with tuna from Japan
In this talk...

- Background
- Dynamic Programming for Incremental Parsing
- Features: from sparse to neural to recurrent neural nets
- Bidirectional RNNs: minimal features; no tree structures!
  - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
  - span-based constituency parsing (Cross+Huang, 2016b)
- Marrying DP & RNNs (mostly not my work!)
  - transition-based dependency parsing (Shi et al, EMNLP 2017)
  - minimal span-based constituency parsing (Stern et al, ACL 2017)
Spectrum: Neural Incremental Parsing

Feedforward NNs
(Chen + Manning 14)

Stack LSTM
(Dyer + 15)

RNNG
(Dyer + 16)

DP incremental parsing
(Huang+Sagae 10, Kuhlmann+ 11)

biRNN dependency
(Kiperwaser+Goldberg 16;
Cross+Huang 16a)

biRNN span-based constituency
(Cross+Huang 16b)

biRNN graph-based dependency
(Kiperwaser+Goldberg 16;
Wang+Chang 16)

minimal span-based constituency
(Stern+ ACL 17)

minimal dependency
(Shi+ EMNLP 17)

edge-factored
(McDonald+ 05a)

all tree info
(summarize output $y$)

DP impossible enables slow DP

enables fast DP

fastest DP: $O(n^3)$

minimal or no tree info
(summarize input $x$)
Spectrum: Neural Incremental Parsing

Feedforward NNs
(Chen + Manning 14)

Stack LSTM
(Dyer+ 15)

RNNG
(Dyer+ 16)

DP incremental parsing
(Huang+Sagae 10, Kuhlmann+ 11)

biRNN dependency
(Kiperwaser+Goldberg 16; Cross+Huang 16a)

biRNN graph-based dependency
(Kiperwaser+Goldberg 16; Wang+Chang 16)

biRNN span-based constituency
(Cross+Huang 16b)

minimal span-based constituency
(Stern+ ACL 17)

minimal dependency
(Shi+ EMNLP 17)

edge-factored
(McDonald+ 05a)

all tree info
(summarize output y)

DP impossible enables slow DP

enables fast DP

fastest DP: $O(n^3)$

minimal or no tree info
(summarize input x)
Incremental Parsing with Dynamic Programming

(Huang & Sagae, ACL 2010*; Kuhlmann et al., ACL 2011; Mi & Huang, ACL 2015)

* best paper nominee
Incremental Parsing with Dynamic Programming

(Huang & Sagae, ACL 2010*; Kuhlmann et al., ACL 2011; Mi & Huang, ACL 2015)
* best paper nominee
Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
</tbody>
</table>
I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
<tr>
<td>2</td>
<td>shift</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td></td>
<td>l eat</td>
<td></td>
</tr>
</tbody>
</table>
I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
<tr>
<td>2</td>
<td>shift</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>3</td>
<td>l-reduce</td>
<td>sushi with tuna ...</td>
</tr>
</tbody>
</table>

Liang Huang (Oregon State)
**Incremental Parsing (Shift-Reduce)**

I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
<tr>
<td>2</td>
<td>shift</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>3</td>
<td>l-reduce</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>4</td>
<td>shift</td>
<td>with tuna from ...</td>
</tr>
</tbody>
</table>
Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
<tr>
<td>2</td>
<td>shift</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>3</td>
<td>l-reduce</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>4</td>
<td>shift</td>
<td>with tuna from ...</td>
</tr>
<tr>
<td>5a</td>
<td>r-reduce</td>
<td>with tuna from ...</td>
</tr>
</tbody>
</table>
Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
<tr>
<td>2</td>
<td>shift</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>3</td>
<td>l-reduce</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>4</td>
<td>shift</td>
<td>with tuna from ...</td>
</tr>
<tr>
<td>5a</td>
<td>r-reduce</td>
<td>with tuna from ...</td>
</tr>
<tr>
<td>5b</td>
<td>shift</td>
<td>tuna from Japan ...</td>
</tr>
</tbody>
</table>
Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

<table>
<thead>
<tr>
<th>action</th>
<th>stack</th>
<th>queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>I eat sushi ...</td>
</tr>
<tr>
<td>1</td>
<td>shift</td>
<td>eat sushi with ...</td>
</tr>
<tr>
<td>2</td>
<td>shift</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>3</td>
<td>l-reduce</td>
<td>sushi with tuna ...</td>
</tr>
<tr>
<td>4</td>
<td>shift</td>
<td>with tuna from ...</td>
</tr>
<tr>
<td>5a</td>
<td>r-reduce</td>
<td>with tuna from ...</td>
</tr>
<tr>
<td>5b</td>
<td>shift</td>
<td>tuna from Japan ...</td>
</tr>
</tbody>
</table>

shift-reduce conflict
Greedy Search

- each state => three new states (shift, l-reduce, r-reduce)
- greedy search: always pick the best next state
  - “best” is defined by a score learned from data
Greedy Search

- each state => three new states (shift, l-reduce, r-reduce)
- greedy search: always pick the best next state
  - “best” is defined by a score learned from data
Beam Search

- each state => three new states (shift, l-reduce, r-reduce)
- beam search: always keep top-$b$ states
  - still just a tiny fraction of the whole search space
Beam Search

- each state => three new states (shift, l-reduce, r-reduce)
- beam search: always keep top-\(b\) states
- still just a tiny fraction of the whole search space

psycholinguistic evidence: parallelism (Fodor et al, 1974; Gibson, 1991)
Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space
Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

(Huang and Sagae, 2010)
Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

(Huang and Sagae, 2010)
Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

```
each DP state corresponds to exponentially many non-DP states
```

```
graph-structured stack
(Tomita, 1986)
```

(Huang and Sagae, 2010)
Dynamic Programming

- each state $\Rightarrow$ three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
- merge equivalent states $\Rightarrow$ polynomial space

Each DP state corresponds to exponentially many non-DP states

Number of trees explored vs. sentence length: DP: exponential vs. non-DP beam search (Huang and Sagae, 2010)
Dynamic Programming

- each state $\Rightarrow$ three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states $\Rightarrow$ polynomial space

Each DP state corresponds to exponentially many non-DP states

Graph-structured stack
(Tomita, 1986)

Number of trees explored vs. sentence length

DP: exponential

Non-DP beam search

(Huang and Sagae, 2010)
Merging (Ambiguity Packing)

- two states are equivalent if they agree on features
- because same features guarantee same cost
- example: if we only care about the last 2 words on stack

(Huang and Sagae, 2010)
Merging (Ambiguity Packing)

- two states are equivalent if they agree on features
- because same features guarantee same cost
- example: if we only care about the last 2 words on stack

(Huang and Sagae, 2010)
Merging (Ambiguity Packing)

- two states are equivalent if they agree on features
- because same features guarantee same cost
- example: if we only care about the last 2 words on stack

psycholinguistic evidence (eye-tracking experiments):

**delayed disambiguation**

John and Mary had 2 papers

Frazier and Rayner (1990), Frazier (1999)
Merging (Ambiguity Packing)

- two states are equivalent if they agree on features
- because same features guarantee same cost
- example: if we only care about the last 2 words on stack

psycholinguistic evidence (eye-tracking experiments):

**delayed disambiguation**

John and Mary had 2 papers each
John and Mary had 2 papers together

Frazier and Rayner (1990), Frazier (1999)
Result: linear-time, DP, and accurate!

- very fast linear-time dynamic programming parser
- explores *exponentially* many trees (and outputs forest)
- state-of-the-art parsing accuracy on English & Chinese
Result: linear-time, DP, and accurate!

- very fast linear-time dynamic programming parser
- explores *exponentially* many trees (and outputs forest)
- state-of-the-art parsing accuracy on English & Chinese
Result: linear-time, DP, and accurate!

- very fast linear-time dynamic programming parser
- explores \textit{exponentially} many trees (and outputs forest)
- state-of-the-art parsing accuracy on English & Chinese
In this talk...

- Background
- Dynamic Programming for Incremental Parsing
  - Features: from sparse to neural to recurrent neural nets
  - Bidirectional RNNs: minimal features; no tree structures!
    - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
    - span-based constituency parsing (Cross+Huang, 2016b)
  - Marrying DP & RNNs (*mostly not my work!*)
    - minimal span-based constituency parsing (Stern et al, ACL 2017)
    - transition-based dependency parsing (Shi et al, EMNLP 2017)
Sparse Features

• score each action using features $f$ and weights $w$
  • features are drawn from a local window
    • abstraction (or signature) of a state -- this inspires DP!
  • weights trained by structured perceptron (Collins 02)
Sparse Features

- score each action using features $f$ and weights $w$
  - features are drawn from a *local window*
  - abstraction (or signature) of a state -- this inspires DP!
  - weights trained by structured perceptron (Collins 02)

(Huang+Sagae, 2010)
Sparse Features

- score each action using features $\mathbf{f}$ and weights $\mathbf{w}$
- features are drawn from a local window
- abstraction (or signature) of a state -- this inspires DP!
- weights trained by structured perceptron (Collins 02)

features:
$$(s_0.w, s_0.rc, q_0, \ldots) = (\text{cats, nearby, in, \ldots})$$

(Huang+Sagae, 2010)
From Sparse to Neural to RNN

(Chen+Manning 2014)
From Sparse to Neural to RNN

- neural nets can automate feature engineering :-)  
- but early neural work (e.g., Chen+Manning 14) still use lots of manually designed atomic features on the stack
neural nets can automate feature engineering :-)  
but early neural work (e.g., Chen+Manning 14) still use lots of manually designed atomic features on the stack  

- option 1: summarize the whole stack (part of $y$) using RNNs => stack LSTM / RNNG (Dyer+ 15, 16)  
- option 2: summarize the whole input ($x$) using RNNs => biLSTM dependency parsing (Kipperwaser+Goldberg 16, Cross+Huang 16a) biLSTM constituency parsing (Cross+Huang 16b)
neural nets can automate feature engineering :-)  
but early neural work (e.g., Chen+Manning 14) still use lots of manually designed atomic features on the stack  

option 1: summarize **the whole stack (part of y)** using RNNs =>  
stack LSTM / RNNG (Dyer+ 15, 16)  
**rules out DP! :(**

option 2: summarize **the whole input (x)** using RNNs =>  
biLSTM dependency parsing (Kiperwaser+Goldberg 16, Cross+Huang 16a)  
biLSTM constituency parsing (Cross+Huang 16b)  
**enables DP! :)**
Spectrum: Neural Incremental Parsing

Feedforward NNs
(Chen + Manning 14)

Stack LSTM
(Dyer + 15)

RNNG
(Dyer + 16)

DP incremental parsing
(Huang+Sagae 10, Kuhlmann+ 11)

biRNN dependency
(Kiperwaser+Goldberg 16; Cross+Huang 16a)

biRNN span-based
countituency
(Cross+Huang 16b)

edge-factored
(McDonald+ 05a)

DP impossible enables slow DP
enables fast DP
fastest DP: \(O(n^3)\)

all tree info
(summarize output \(y\))

minimal or no tree info
(summarize input \(x\))

minimal dependency
(Shi+ EMNLP 17)

constituency
dependency
bottom-up

biRNN graph-based
dependency
(Kiperwaser+Goldberg 16; Wang+Chang 16)

minimal span-based
countituency
(Cross+ACL 17)
In this talk...

• Background

• Dynamic Programming for Incremental Parsing

• Interlude: NN Features: from feedforward to recurrent
  • Bidirectional RNNs: minimal features; no tree structures!
    • dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
    • span-based constituency parsing (Cross+Huang, 2016b)
  • Marrying DP & RNNs *(mostly not my work!)*
    • minimal span-based constituency parsing (Stern et al, ACL 2017)
    • transition-based dependency parsing (Shi et al, EMNLP 2017)
biRNN for Dependency Parsing

• several parallel efforts in 2016 used biLSTM features
  • Kiperwaser+Goldberg 2016: four positional feats; arc-eager
  • Cross+Huang ACL 2016: three positional feats; arc-standard
  • Wang+Chang 2016: two positional feats; graph-based
• all inspired by sparse edge-factored model (McDonald+05)
  • use positions to summarize the input $\mathbf{x}$, not the output $\mathbf{y}$!
  • $\Rightarrow O(n^3)$ DP, e.g. graph-based, but also incremental!

these developments lead to state-of-the-art in dependency parsing
Span-Based Constituency Parsing

• previous work uses tree structures on stack
• we simplify to operate directly on sentence spans
• simple-to-implement linear-time parsing

previous work

our work

(Cross and Huang, EMNLP 2016)
<table>
<thead>
<tr>
<th>Structural (even step)</th>
<th>Shift</th>
<th>Combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label (odd step)</td>
<td>Label-X</td>
<td>No-Label</td>
</tr>
</tbody>
</table>

- **Structural (even step)**
  - Shift
  - Combine

- **Label (odd step)**
  - Label-X
  - No-Label

![Diagram of a sentence tree and brackets](image)

```
I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
```

Current brackets: \( t = \{ \} \)

(Cross and Huang, EMNLP 2016)
Structural (even step)

- Shift
- Combine

Label (odd step)

- Label-X
- No-Label

(Cross and Huang, EMNLP 2016)
Structural (even step)

<table>
<thead>
<tr>
<th>Shift</th>
<th>Combine</th>
</tr>
</thead>
</table>

Label (odd step)

<table>
<thead>
<tr>
<th>Label-X</th>
<th>No-Label</th>
</tr>
</thead>
</table>

- **Structural (even step)**
  - Shift
  - Combine

- **Label (odd step)**
  - Label-X
  - No-Label

```
I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
```

- **Shift**

```
I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
```

- **Label-NP**

```
I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
```

```
NP  | VP
---|---
PRP  | MD  VBP
I  | do  like S
NP  | eating NN

VBP  | NN
```

- **current brackets**

```
t = {}
```

- **t = \{NP\}**
Liang Huang (Oregon State)

Structural (even step)
- Shift
- Combine

Label (odd step)
- Label-X
- No-Label

(Cross and Huang, EMNLP 2016)
Structural (even step)
- Shift
- Combine

Label (odd step)
- Label-X
- No-Label

\[ t = \{ \} \]

Structural (even step)
- Shift
- Combine

Label-NP
\[ t = \{0NP_1\} \]

No-Label
\[ t = \{0NP_1\} \]
Structural (even step)

- Shift
- Combine

Label (odd step)

- Label-X
- No-Label

(Cross and Huang, EMNLP 2016)
Structural (even step)
- Shift
- Combine
Label (odd step)
- Label-X
- No-Label

(Cross and Huang, EMNLP 2016)
Structural (even step)

- Shift
- Combine

Label (odd step)

- Label-X
- No-Label

Cross and Huang, EMNLP 2016

$\text{I/PRP} \quad \text{do/MD} \quad \text{like/VBP} \quad \text{eating/VBG} \quad \text{fish/NN} \quad t = \{_0\text{NP}_1\}$
Structural (even step)
Shift
Combine
Label (odd step)
Label-X
No-Label

I/PRP  do/MD  like/VBP
0 1 2

eating/VBG  fish/NN
3 4 5

Combine

I/PRP  do/MD  like/VBP
0 1 2

eating/VBG  fish/NN
3 4 5

t = \{_0\text{NP}_1\}

(Cross and Huang, EMNLP 2016)
Structural (even step) | Shift
---|---
| Combine
Label (odd step) | Label-X
| No-Label

(Cross and Huang, EMNLP 2016)
<table>
<thead>
<tr>
<th>Structural (even step)</th>
<th>Shift</th>
<th>Combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label (odd step)</td>
<td>Label-X</td>
<td>No-Label</td>
</tr>
</tbody>
</table>

Cross and Huang, EMNLP 2016
Structural (even step)

Shift

Combine

Label (odd step)

Label-X

No-Label

(t = \{0\})

(t = \{0\})

(t = \{0\})

(Cross and Huang, EMNLP 2016)
Structural (even step)

Combine

Label (odd step)

Label-

No-Label

(t = \{0_{NP}\})

I/PRP do/MD like/VBP eating/VBG fish/NN

(t = \{0_{NP}\})

I/PRP do/MD like/VBP eating/VBG fish/NN

(t = \{0_{NP}\})

I/PRP do/MD like/VBP eating/VBG fish/NN

(t = \{0_{NP}\})

I/PRP do/MD like/VBP eating/VBG fish/NN

(Cross and Huang, EMNLP 2016)
Structural (even step)

Combine

Label (odd step)

Label-NP

No-Label

I/PRP do/MD like/VBP eating/VBG fish/NN

\[ t = \{0NP_1\} \]

Combine

I/PRP do/MD like/VBP eating/VBG fish/NN

\[ t = \{0NP_1\} \]

Shift

I/PRP do/MD like/VBP eating/VBG fish/NN

\[ t = \{0NP_1\} \]

Shift

I/PRP do/MD like/VBP eating/VBG fish/NN

\[ t = \{0NP_1, 4NP_5\} \]
Structural (even step)  
Shift  
Combine  

Label (odd step)  
Label-X  
No-Label  

I/PRP  
do/MD  
like/VBP  
eating/VBG  
fish/NN

S  
NP  
PRP  
I  
VP  
MD  
do  
like  
NP  
eating  
NN  
fish

t = \{I_{NP_1}, 4_{NP_5}\}

(Cross and Huang, EMNLP 2016)
 Structural (even step)  
<table>
<thead>
<tr>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combine</td>
</tr>
</tbody>
</table>

 Label (odd step)  
| Label-X |
| No-Label |

I/PRP  | do/MD  | like/VBP  | eating/VBG  | fish/NN |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

↓ Combine

I/PRP  | do/MD  | like/VBP  | eating/VBG  | fish/NN |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

S → VP  
NP  
| PRP |
I

VP  
MD  
do  
like

S  
NP  
eating  
NN  
fish

t = \{6NP_1, 4NP_5\}  
(Cross and Huang, EMNLP 2016)
Structural (even step)

Shift

Combine

Label (odd step)

Label-X

No-Label

I/PRP 0 do/MD 1 like/VBP 3 eating/VBG 4 fish/NN 5

Combine

I/PRP 0 do/MD 1 like/VBP 3 eating/VBG 4 fish/NN 5

Label-S-VP

S

VP

NP

PRP

I

MD

VBP

do

like

S

VP

VBG

NP

eating

NN

fish

\[ t = \{0, 4\} \]

\[ t = \{0, 4, S_5, V_5\} \]
Structural (even step)
- Shift
- Combine

Label (odd step)
- Label-X
- No-Label

**Example:**

I/PRP do/MD like/VBP eating/VBG fish/NN

\[ t = \{\text{NP}, \text{NP}_5\} \]

**Combine**

I/PRP do/MD like/VBP eating/VBG fish/NN

\[ t = \{\text{NP}, \text{NP}_5, \text{S}_5, \text{VP}_5\} \]

**Label-S-VP**
Structural (even step)

Shift

Combine

Label (odd step)

Label-X

No-Label

Liang Huang (Oregon State)  (Cross and Huang, EMNLP 2016)
Liang Huang (Oregon State)

Structural (even step)
- Shift
- Combine

Label (odd step)
- Label-X
- No-Label

(Cross and Huang, EMNLP 2016)
Structural (even step)  
- Shift
- Combine

Label (odd step)  
- Label-X
- No-Label

Diagram illustrating the process of parsing a sentence with structural and label operations. The sentence is: "I do like eating fish.

Example steps:
1. Initial state: I/PRP 0, do/MD 1, like/VBP 2, eating/VBG 3, fish/NN 4
2. Structural: Shift
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
3. Label: Label-X
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
4. Structural: Shift
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
5. Label: Label-S-VP
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
6. Structural: Shift
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
7. Label: Label-VP
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
8. Structural: Shift
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5
9. Label: Label-S
   - Combined state: I/PRP 0, do/MD 1, like/VBP 3, eating/VBG 4, fish/NN 5

The final state represents the parsed sentence with labels and structure.
Bi-LSTM Span Features

- Sentence segment “eating fish” represented by two vectors:
  - Forward component: $f_5 - f_3$ (Wang and Chang, ACL 2016)
  - Backward component: $b_3 - b_5$ (Cross and Huang, EMNLP 2016)
Structural Action: 4 spans

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN

Label Action: 3 spans

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
## Results on Penn Treebank

<table>
<thead>
<tr>
<th>Parser</th>
<th>Search</th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carreras et al. (2008)</td>
<td>cubic</td>
<td>90.7</td>
<td>91.4</td>
<td>91.1</td>
</tr>
<tr>
<td>Shindo et al. (2012)</td>
<td>cubic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thang et al. (2015)</td>
<td>~cubic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Watanabe et al. (2015)</td>
<td>beam</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Static Oracle</strong></td>
<td>greedy</td>
<td>90.7</td>
<td>91.4</td>
<td>91.0</td>
</tr>
<tr>
<td><strong>Dynamic + Exploration</strong></td>
<td>greedy</td>
<td>90.5</td>
<td>92.1</td>
<td><strong>91.3</strong></td>
</tr>
</tbody>
</table>

- state of the art despite simple system with greedy actions and small embeddings trained from scratch
- first neural constituency parser to outperform sparse features

(Cross and Huang, EMNLP 2016)
Extension: Joint Syntax-Discourse Parsing

- extend span-based parsing to discourse parsing
- end-to-end, joint syntactic and discourse parsing

(Kai and Huang, EMNLP 2017)
In this talk...

- **Background**
- **Dynamic Programming for Incremental Parsing**
- **Interlude: NN Features: from feedforward to recurrent**
- **Bidirectional RNNs: minimal features; no tree structures!**
  - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
  - span-based constituency parsing (Cross+Huang, 2016b)
- **Marrying DP & RNNs** (*mostly not my work!*)
  - minimal span-based constituency parsing (Stern et al, ACL 2017)
  - transition-based dependency parsing (Shi et al, EMNLP 2017)
Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
- an even simpler score formulation
- $O(n^3)$ exact DP (CKY) instead of greedy search
- global loss-augmented training instead of local training
Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
- an even simpler score formulation
- $O(n^3)$ exact DP (CKY) instead of greedy search
- global loss-augmented training instead of local training

(Cross+Huang, EMNLP16)
Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
- an even simpler score formulation
- $O(n^3)$ exact DP (CKY) instead of greedy search
- global loss-augmented training instead of local training

(Cross+Huang, EMNLP16)

- structural action
  \[ \text{score}_{\text{action}}(i, k, j) \]

- label action
  \[ \text{score}_{\text{label}}(i, j) \]
Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
- an even simpler score formulation
- $O(n^3)$ exact DP (CKY) instead of greedy search
- global loss-augmented training instead of local training

(Cross+Huang, EMNLP16)  (Stern+, ACL 2017)

structural action

\[
\text{score}_{\text{action}}(i, k, j) = \max_k \text{best } (i, k) + \text{best } (k, j)
\]

label action

\[
\text{score}_{\text{label}}(i, j) = \max_{\text{label}} \text{score}_{\text{label}}(i, j)
\]
Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
- an even simpler score formulation
- $O(n^3)$ exact DP (CKY) instead of greedy search
- global loss-augmented training instead of local training

(Cross+Huang, EMNLP16)

structural action

$$\text{score}_{\text{action}}(i, k, j)$$

$$\max_k \text{best} (i, k) + \text{best} (k, j)$$

(Stern+, ACL 2017)

label action

$$\text{score}_{\text{label}}(i, j)$$

$$\max_{\text{label}} \text{score}_{\text{label}}(i, j)$$
Global Training & Loss-Augmented Decoding

want \( s_{tree}(T^*) > s_{tree}(T) \) for all \( T \neq T^* \)

and larger margin for worse trees: \( s_{tree}(T^*) \geq \Delta(T, T^*) + s_{tree}(T) \)

loss-augmented decoding in training (find the most-violated tree, i.e., a \textit{bad tree} with \textit{good score})

\[
\hat{T} = \max_{T} [\Delta(T, T^*) + s_{tree}(T)]
\]

\textit{bad tree} \quad \textit{good score}

loss-augmented decoding for Hamming loss (approximating F1):

simply replace score \( \text{label}(i, j) \)

with score \( \text{label}(i, j) + 1(\text{label} \neq \text{label}_{ij}^*) \)

\( \text{gold tree label for span } (i, j) \)

(could be “nolabel”)

(Stern+, ACL 2017)
<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall et al. (2014)</td>
<td>89.2</td>
</tr>
<tr>
<td>Vinyals et al. (2015)</td>
<td>88.3</td>
</tr>
<tr>
<td>Cross and Huang (2016b)</td>
<td>91.3</td>
</tr>
<tr>
<td>Dyer et al. (2016) corrected</td>
<td>91.7</td>
</tr>
<tr>
<td>Liu and Zhang (2017)</td>
<td>91.7</td>
</tr>
<tr>
<td>Chart Parser</td>
<td>91.7</td>
</tr>
<tr>
<td>+refinement</td>
<td>91.8</td>
</tr>
</tbody>
</table>

(Stern+, ACL 2017)
Minimal Feats for Incremental Dep. Parsing

(Kiperwaser and Goldberg 2016)
arc-eager

(Cross and Huang, ACL 2016)
arc-standard
Minimal Feats for Incremental Dep. Parsing

(Kiperwaser and Goldberg 2016)
arc-eager

(Cross and Huang, ACL 2016)
arc-standard

(Shi, Huang, Lee, EMNLP 2017)
Saturday talk!
arc-hybrid and arc-eager

works for both greedy and $O(n^3)$ DP
Minimal Feats for Incremental Dep. Parsing
Spectrum: Neural Incremental Parsing

Feedforward NNs
(Chen + Manning 14)

Stack LSTM
(Dyer + 15)

RNNG
(Dyer + 16)

DP incremental parsing
(Huang+Sagae 10, Kuhlmann+ 11)

DP impossible
enables slow DP

edge-factored
(McDonald+ 05a)

biRNN dependency
(Kiperwaser+Goldberg 16; Cross+Huang 16a)

biRNN graph-based dependency
(Kiperwaser+Goldberg 16; Wang+Chang 16)

biRNN span-based constituency
(Cross+Huang 16b)

minimal span-based constituency
(Stern+ ACL 17)

minimal dependency
(Shi+ EMNLP 17)

all tree info
(summarize output y)

DP impossible
enables fast DP

minimal or no tree info
(summarize input x)

fastest DP: $O(n^3)$
Spectrum: Neural Incremental Parsing

Feedforward NNs
(Chen + Manning 14)

Stack LSTM
(Dyer+ 15)

RNNG
(Dyer+ 16)

DP incremental parsing
(Huang+Sagae 10, Kuhlmann+ 11)

biRNN dependency
(Kiperwaser+Goldberg 16; Cross+Huang 16a)

biRNN span-based constituency
(Cross+Huang 16b)

edge-factored
(McDonald+ 05a)

biRNN graph-based dependency
(Kiperwaser+Goldberg 16; Wang+Chang 16)

minimal span-based constituency
(Stern+ ACL 17)

minimal dependency
(Shi+ EMNLP 17)

all tree info
(summarize output y)

DP impossible enables slow DP

enables fast DP

minimal or no tree info
(summarize input x)

fastest DP: O(n^3)
Conclusions and Limitations
Conclusions and Limitations

- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input $\mathbf{x}$ and not output structure $\mathbf{y}$
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)
Conclusions and Limitations

- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input $\mathbf{x}$ and not output structure $\mathbf{y}$
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)
- but exact DP is still too slow
  - future work: linear-time beam search DP with biRNNs
Conclusions and Limitations

• DP and RNNs can indeed be married, if done creatively
  • biRNN summarizing input \( x \) and not output structure \( y \)
  • this allows efficient DP with exact search
  • combine with global learning (loss-augmented decoding)

• but exact DP is still too slow
  • future work: linear-time beam search DP with biRNNs

• what if we want strictly incremental parsing? no biRNN...
  • DP search could compensate for loss of lookahead
Conclusions and Limitations

- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input $\mathbf{x}$ and not output structure $\mathbf{y}$
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)
- but exact DP is still too slow
  - future work: linear-time beam search DP with biRNNs
- what if we want strictly incremental parsing? no biRNN...
  - DP search could compensate for loss of lookahead
- what about translation? we do need to model $\mathbf{y}$ directly...
非常 感谢！
fēi cháng  gǎn xiè
非常 感谢！

fēi cháng  gǎn xiè

Thank you very much!