Part 6: Applications of Structure
Shallow Learning

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

Sentence
Deep Learning

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

End-to-End

Sentence
A Question

• Is Parsing or Structure Necessary?

<table>
<thead>
<tr>
<th>Task</th>
<th>Bi-LSTM</th>
<th>Tree-LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford Sentiment TreeBank</td>
<td>49.8 / 50.7</td>
<td>50.4</td>
</tr>
<tr>
<td>Binary Sentiment Classification</td>
<td>79.0</td>
<td>77.4</td>
</tr>
<tr>
<td>Question-Answer Matching</td>
<td>56.4</td>
<td>55.8</td>
</tr>
<tr>
<td>Semantic Relationship Classification</td>
<td>75.2</td>
<td>76.7</td>
</tr>
<tr>
<td>Discourse Parsing</td>
<td>57.5</td>
<td>56.4</td>
</tr>
</tbody>
</table>

How to Use Tree or Graph Structures?

- As Information Extraction Rules
- As Input Features
- As Input Structures
- As Structured Prediction
How to Use Tree or Graph Structures?

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

• For example
  – Polarity-target pair extraction

• Problem
  – The extraction rules are very complex
  – The parsing results are inexact
As Information Extraction Rules

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

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

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

• As Information Extraction Rules
• As Input Features
• As Input Structures
• As Structured Prediction
Path Features

• For Example
  – Semantic Role Labeling (SRL), Relation Extraction (RC)

  ![Diagram](image)

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

• The parsing path features are very important
  – People <-- downtown: nsubj ← moved → nmod
• But they are difficult to be designed and very sparse
Path Features

- Use LSTMs to represent paths
- All of word, POS tags and relations can be inputted

Michael Roth and Mirella Lapata. Neural Semantic Role Labeling with Dependency Path Embeddings. ACL 2016.
Joint learning of SRL and RC

• Multi-task learning

Hidden Units of Parsing as Features

- The hidden units for parsing include **soft** syntactic information.
- These can help applications, such as relation extraction.

---

How to Use Tree or Graph Structures?

• As Information Extraction Rules
• As Input Features
• As Input Structures
• As Structured Prediction
Recurrent vs. Recursive Neural Networks

- Recurrent Neural Networks
  - Composing sequentially

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

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng and Christopher D. Manning. Parsing Natural Scenes And Natural Language With Recursive Neural Networks. ICML 2011.
Tree-LSTMs

- Standard LSTM

- Tree-LSTM

Tree-LSTMs

\[ h_j = \sum_{k \in C(j)} h_k, \]
\[ i_j = \sigma \left( W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)} \right), \]
\[ f_{jk} = \sigma \left( W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right), \]
\[ o_j = \sigma \left( W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)} \right), \]
\[ u_j = \tanh \left( W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)} \right), \]
\[ c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \]
\[ h_j = o_j \odot \tanh(c_j), \]

All patients were treated with gefitinib and showed a partial response.

Neural Machine Translation

How to Use Tree or Graph Structures?

• As Information Extraction Rules
• As Input Features
• As Input Structures
• As Structured Prediction
Event Extraction

• Event Extraction as Dependency Parsing

Disfluency Detection

• Disfluency detection for speech recognition

  I want a flight [ to Boston + \{um\} to Denver ]

  RM IM RP

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

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

• An Example of transition-based disfluency detection

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Output</th>
<th>Stack</th>
<th>Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OUT</td>
<td>[]</td>
<td>[]</td>
<td>[a, flight, to, boston, to, denver]</td>
</tr>
<tr>
<td>1</td>
<td>OUT</td>
<td>[a]</td>
<td>[]</td>
<td>[flight, to, boston, to, denver]</td>
</tr>
<tr>
<td>2</td>
<td>OUT</td>
<td>[a, flight]</td>
<td>[]</td>
<td>[to, boston, to, denver]</td>
</tr>
<tr>
<td>3</td>
<td>DEL</td>
<td>[a, flight]</td>
<td>[to]</td>
<td>[boston, to, denver]</td>
</tr>
<tr>
<td>4</td>
<td>DEL</td>
<td>[a, flight]</td>
<td>[to, boston]</td>
<td>[to, denver]</td>
</tr>
<tr>
<td>5</td>
<td>OUT</td>
<td>[a, flight, to]</td>
<td>[]</td>
<td>[denver]</td>
</tr>
<tr>
<td>6</td>
<td>OUT</td>
<td>[a, flight, to, denver]</td>
<td>[]</td>
<td>[]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our</td>
<td>91.1</td>
<td>84.1</td>
<td>87.5</td>
</tr>
<tr>
<td>Attention-based (Wang et al., 2016)</td>
<td>91.6</td>
<td>82.3</td>
<td>86.7</td>
</tr>
<tr>
<td>Bi-LSTM (Zayats et al., 2016)</td>
<td>91.8</td>
<td>80.6</td>
<td>85.9</td>
</tr>
<tr>
<td>semi-CRF (Ferguson et al., 2015)</td>
<td>90.0</td>
<td>81.2</td>
<td>85.4</td>
</tr>
<tr>
<td>UBT (Wu et al., 2015)</td>
<td>90.3</td>
<td>80.5</td>
<td>85.1</td>
</tr>
<tr>
<td>M^3N (Qian and Liu, 2013)</td>
<td>-</td>
<td>-</td>
<td>84.1</td>
</tr>
</tbody>
</table>

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

• As Information Extraction Rules
• As Input Features
• As Input Structures
• As Structured Prediction
Course Summarization

• Lexical, Syntactic and Semantic Analysis
  – Structured Prediction (Segmentation, Tagging and Parsing)

• Deep Learning
  – Representation Learning
  – End-to-end Learning

• Traditional Methods
  – Graph-based and Transition-based

• Neural Network Methods
  – Graph-based and Transition-based

• Applications