Policy Gradient as a Proxy for Dynamic Oracles in Constituency Parsing

Berkeley

Daniel Fried and Dan Klein
The cat took a nap.

$$L(\theta) = \log p(y|x; \theta) = \sum_t \log p(y_t|y_{1:t-1}, x; \theta)$$
Non-local Consequences

Loss-Evaluation Mismatch

\[ \Delta(y, \hat{y}) : -F1(y, \hat{y}) \]

Exposure Bias

True Parse

\[ y \rightarrow (S) \rightarrow (NP) \rightarrow \text{The} \rightarrow \text{cat} \rightarrow \ldots \]

Prediction

\[ \hat{y} \rightarrow (S) \rightarrow (NP) \rightarrow (VP) \rightarrow \ldots \]

[Ranzato et al. 2016; Wiseman and Rush 2016]
Dynamic Oracle Training

Explore at training time. Supervise each state with an expert policy.

True Parse $y$: $y = (S \rightarrow (NP \rightarrow \text{The} \rightarrow \text{cat} \rightarrow ...$

Prediction
- (sample, or greedy)

Oracle
- $y^*$

$L(\theta) = \sum_t \log p(y^*_t | \hat{y}_{1:t-1}, x; \theta)$

choose $y^*_t$ to maximize achievable F1 (typically)

addresses loss mismatch

addresses exposure bias

[Goldberg & Nivre 2012; Ballesteros et al. 2016; inter alia]
**Dynamic Oracles Help!**

**Expert Policies / Dynamic Oracles**
- Daume III et al., 2009; Ross et al., 2011;
- Choi and Palmer, 2011; Goldberg and Nivre, 2012;
- Chang et al., 2015; Ballesteros et al., 2016; Stern et al. 2017

**PTB Constituency Parsing F1**

<table>
<thead>
<tr>
<th>System</th>
<th>Static Oracle</th>
<th>Dynamic Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coavoux and Crabbé, 2016</td>
<td>88.6</td>
<td>89.0</td>
</tr>
<tr>
<td>Cross and Huang, 2016</td>
<td>91.0</td>
<td>91.3</td>
</tr>
<tr>
<td>Fernández-González and Gómez-Rodríguez, 2018</td>
<td>91.5</td>
<td>91.7</td>
</tr>
</tbody>
</table>

mostly dependency parsing
What if we don’t have a dynamic oracle?  

*Use reinforcement learning*
Reinforcement Learning Helps! (in other tasks)

Auli and Gao, 2014; Ranzato et al., 2016; Shen et al., 2016
Xu et al., 2016; Wiseman and Rush, 2016; Edunov et al. 2017

- CCG parsing
- several, including dependency parsing
- machine translation
Policy Gradient Training

Minimize expected sequence-level cost:

\[ R(\theta) = \sum_{\hat{y}} p(\hat{y}|x; \theta) \Delta(y, \hat{y}) \]

\[ \nabla R(\theta) = \sum_{\hat{y}} p(\hat{y}|x; \theta) \Delta(y, \hat{y}) \nabla \log p(\hat{y}|x; \theta) \]

addresses exposure bias (compute by sampling)
addresses loss (compute F1)
compute in the same way as for the true tree

[Williams, 1992]
Policy Gradient Training

\[ \nabla R(\theta) = \sum_{\hat{y}} p(\hat{y}|x; \theta) \Delta(y, \hat{y}) \nabla \log p(\hat{y}|x; \theta) \]

Input, \( x \)

The cat took a nap.

\( k \) candidates, \( \hat{y} \)

\( \Delta(y, \hat{y}) \) (negative F1)

gradient for candidate

\[ \nabla \log p(\hat{y}_1|x; \theta) \quad \nabla \log p(\hat{y}_2|x; \theta) \quad \nabla \log p(\hat{y}_3|x; \theta) \quad \nabla \log p(y|x; \theta) \]
Experiments
Setup

**Parsers**
- Span-Based [Cross & Huang, 2016]
- Top-Down [Stern et al. 2016]
- RNNG [Dyer et al. 2016]
- In-Order [Liu and Zhang, 2017]

**Training**
- Static oracle
- Dynamic oracle
- Policy gradient

X
Training Efficiency

PTB learning curves for the Top-Down parser

- Development F1 vs Training Epoch
- Static oracle, Dynamic oracle, Policy gradient
Chinese Penn Treebank v5.1 F1

- Static oracle
- Policy gradient
- Dynamic oracle

- Span-Based
- Top-Down
- RNNG-128
- RNNG-256
- In-Order
Conclusions

- Local decisions can have non-local consequences
  - Loss mismatch
  - Exposure bias

- How to deal with the issues caused by local decisions?
  - Dynamic oracles: efficient, model specific
  - Policy gradient: slower to train, but general purpose
Thank you!
For Comparison: A Novel Oracle for RNNG

1. Close current constituent if it’s a true constituent...

(S) (NP) The man ) (VP) had ...

... or it could never be a true constituent.

(S) (VP) (NP) The man ) )

2. Otherwise, open the outermost unopened true constituent at this position.

(S) (NP) The man ) (VP)

3. Otherwise, shift the next word.

(S) (NP) The man ) (VP) had