A Generative Parser with a Discriminative Recognition Algorithm

Jianpeng Cheng, Adam Lopez and Mirella Lapata

1st Aug 2017
School of Informatics, University of Edinburgh
Introduction
Comparison between Generative and Discriminative Parsing Models

**Generative parsing models**

- Generative models (e.g., PCFG-based (Collins, 1997)) learn a joint distribution $p(x, y)$ on sentence($x$)—parse tree($y$) pairs.
Comparison between Generative and Discriminative Parsing Models

Generative parsing models

- Generative models (e.g., PCFG-based (Collins, 1997)) learn a joint distribution $p(x, y)$ on sentence($x$)—parse tree($y$) pairs.
- During parsing, it finds the most likely parse tree for a given sentence $\arg\max_y p(y|x)$ (with a chart-based recognition algorithm).
Comparison between Generative and Discriminative Parsing Models

Discriminative parsing models

- Discriminative models (e.g., shift-reduce parser (Ratnaparkhi, 1998)) learn a conditional distribution $p(y|x)$ of parse trees given sentences.
Discriminative parsing models

- Discriminative models (e.g., shift-reduce parser (Ratnaparkhi, 1998)) learn a conditional distribution $p(y|x)$ of parse trees given sentences.
- During parsing, it finds the most likely parse tree for a given sentence $\text{argmax}_y p(y|x)$. 
## Comparison between Generative and Discriminative Parsing Models

<table>
<thead>
<tr>
<th>Feature</th>
<th>Generative</th>
<th>Discriminative</th>
</tr>
</thead>
<tbody>
<tr>
<td>models data generation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>supports both parsing and language modeling</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>consistent training and test objectives</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>supports arbitrary global features (e.g., sentential features)</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>supports linear-time recognition algorithm</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
## Our Work

<table>
<thead>
<tr>
<th></th>
<th>Generative</th>
<th>Discriminative</th>
<th><strong>Our work</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>models data generation</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>supports both parsing and language modeling</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>consistent training and test objectives</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>supports arbitrary global features (e.g., sentential features)</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>supports linear-time recognition algorithm</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Our Work

- Unifies generative and discriminative parsing models;

Showcases in the context of Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016).
Our Work

- Unifies generative and discriminative parsing models;
  - Generative model explains data generation;

Showcases in the context of Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016).
Our Work

- Unifies generative and discriminative parsing models;
  - Generative model explains data generation;
  - Discriminative model does fast linear recognition;
- Training two models with a unified objective;
- Parsing and language modeling within a single framework.

Showcases in the context of Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016).
Our Work

- Unifies generative and discriminative parsing models;
  - Generative model explains data generation;
  - Discriminative model does fast linear recognition;
  - Training two models with a unified objective;

Showcases in the context of Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016).
Our Work

- Unifies generative and discriminative parsing models;
  - Generative model explains data generation;
  - Discriminative model does fast linear recognition;
  - Training two models with a unified objective;
  - Parsing and language modeling within a single framework.

- Showcases in the context of Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016).
Our Work

• Unifies generative and discriminative parsing models;
  • Generative model explains data generation;
  • Discriminative model does fast linear recognition;
  • Training two models with a unified objective;
  • Parsing and language modeling within a single framework.

• Showcases in the context of Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016).
• A neural transition system that generates parse trees with three operations:
RNNG

A neural transition system that generates parse trees with three operations:

- NT(X) (creates a non-terminal tree node);
- SHIFT/GEN (creates a terminal tree node);
- REDUCE (completes a subtree).

Two versions:
- Discriminative RNNG: creates a terminal by shifting it from the input buffer (i.e., with access to the whole sentence)
- Generative RNNG: creates a terminal by generating it conditioned on the generation history (i.e., without access to the whole sentence)
A neural transition system that generates parse trees with three operations:

- $\text{NT}(X)$ (creates a non-terminal tree node);
- $\text{SHIFT/GEN}$ (creates a terminal tree node);
A neural transition system that generates parse trees with three operations:

- **NT(X)** (creates a non-terminal tree node);
- **SHIFT/GEN** (creates a terminal tree node);
- **REDUCE** (completes a subtree).
• A neural transition system that generates parse trees with three operations:
  • \texttt{NT(X)} (creates a non-terminal tree node);
  • \texttt{SHIFT/GEN} (creates a terminal tree node);
  • \texttt{REDUCE} (completes a subtree).

• Two versions:
RNNG

- A neural transition system that generates parse trees with three operations:
  - \textbf{NT}(X) (creates a non-terminal tree node);
  - \textbf{SHIFT/GEN} (creates a terminal tree node);
  - \textbf{REDUCE} (completes a subtree).

- Two versions:
  - \textbf{Discriminative RNNG}: creates a terminal by shifting it from the input buffer (i.e., with access to the whole sentence)
• A neural transition system that generates parse trees with three operations:
  • NT(X) (creates a non-terminal tree node);
  • SHIFT/GEN (creates a terminal tree node);
  • REDUCE (completes a subtree).

• Two versions:
  • Discriminative RNNG: creates a terminal by shifting it from the input buffer (i.e., with access to the whole sentence)
  • Generative RNNG: creates a terminal by generating it conditioned on the generation history (i.e., without access to the whole sentence)
**Figure 1:** Generative RNNG example from Dyer et al. (2016). The input sentence is *The hungry cat meows.*
Methodology
The Proposed Framework

The plate diagram:

- generative RNNG computes the joint $p(x, y) = p(y)p(x|y)$;
The Proposed Framework

The plate diagram:

- generative RNNG computes the joint $p(x, y) = p(y)p(x|y)$;
- discriminative RNNG computes an approximated posterior $q(y|x)$;
The Proposed Framework

The plate diagram:

- generative RNNG computes the joint $p(x, y) = p(y)p(x|y)$;
- discriminative RNNG computes an approximated posterior $q(y|x)$;
- This is a discrete variational autoencoder (Miao and Blunsom, 2016).
Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td></td>
</tr>
</tbody>
</table>
### Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td></td>
</tr>
<tr>
<td>• top embedding on the output buffer (computed by LSTM)</td>
<td></td>
</tr>
</tbody>
</table>
### Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td></td>
</tr>
<tr>
<td>• top embedding on the output buffer (computed by LSTM)</td>
<td></td>
</tr>
<tr>
<td>• parent non-terminal embedding</td>
<td></td>
</tr>
</tbody>
</table>
### Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td>• stack embedding (computed by stack-LSTM)</td>
</tr>
<tr>
<td>• top embedding on the output buffer (computed by LSTM)</td>
<td></td>
</tr>
<tr>
<td>• parent non-terminal embedding</td>
<td></td>
</tr>
</tbody>
</table>
## Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td>• stack embedding (computed by stack-LSTM)</td>
</tr>
<tr>
<td>• top embedding on the output buffer (computed by LSTM)</td>
<td>• top embedding on the input buffer (computed by bidirectional-LSTM)</td>
</tr>
<tr>
<td>• parent non-terminal embedding</td>
<td></td>
</tr>
</tbody>
</table>
### Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td>• stack embedding (computed by stack-LSTM)</td>
</tr>
<tr>
<td>• top embedding on the output buffer (computed by LSTM)</td>
<td>• top embedding on the input buffer (computed by bidirectional-LSTM)</td>
</tr>
<tr>
<td>• parent non-terminal embedding</td>
<td>• buffer embedding (computed by neural attention)</td>
</tr>
</tbody>
</table>
### Features

<table>
<thead>
<tr>
<th>Generative RNNG (decoder)</th>
<th>Discriminative RNNG (encoder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• stack embedding (computed by stack-LSTM)</td>
<td>• stack embedding (computed by stack-LSTM)</td>
</tr>
<tr>
<td>• top embedding on the output buffer (computed by LSTM)</td>
<td>• top embedding on the input buffer (computed by bidirectional-LSTM)</td>
</tr>
<tr>
<td>• parent non-terminal embedding</td>
<td>• buffer embedding (computed by neural attention)</td>
</tr>
<tr>
<td></td>
<td>• parent non-terminal embedding</td>
</tr>
</tbody>
</table>
Training

- Notations: $x$ denotes sentence; $y$ denotes parse tree/action sequence;
generative model computes $p(x, y)$;
discriminative model computes $q(y|x)$
Training

- **Notations:** \(x\) denotes sentence; \(y\) denotes parse tree/action sequence; generative model computes \(p(x, y)\); discriminative model computes \(q(y|x)\).

- **Variational inference objective:** maximize the marginal likelihood of the sentence
  \[ L(vi) = \log p(x) \geq \mathbb{E}_{q(y|x)} \log \frac{p(x, y)}{q(y|x)} \text{ (the variational lower bound)} \]
• Notations: \( x \) denotes sentence; \( y \) denotes parse tree/action sequence; generative model computes \( p(x, y) \); discriminative model computes \( q(y|x) \)

• Variational inference objective: maximize the marginal likelihood of the sentence
  \[
  \mathcal{L}(vi) = \log p(x) \geq \mathbb{E}_{q(y|x)} \log \frac{p(x, y)}{q(y|x)} \quad \text{(the variational lower bound)}
  \]

• Objective for observed parse trees: maximize the conditional likelihood of the parse tree given the sentence
  \[
  \mathcal{L}(cl) = \log p(a|x) + \log q(a|x)
  \]
Training

- Notations: $x$ denotes sentence; $y$ denotes parse tree/action sequence;
generative model computes $p(x, y)$;
discriminative model computes $q(y|x)$

- Variational inference objective: maximize the marginal likelihood of the sentence
  $\mathcal{L}(vi) = \log p(x) \geq \mathbb{E}_{q(y|x)} \log \frac{p(x, y)}{q(y|x)}$ (the variational lower bound)

- Objective for observed parse trees: maximize the conditional likelihood of the parse tree given the sentence
  $\mathcal{L}(cl) = \log p(a|x) + \log q(a|x)$

- We use a combined objective: $\mathcal{L}(vi) + \mathcal{L}(cl)$
Two approaches:

1. Find the action sequence that maximizes the approximated posterior $q(y|x)$;
Two approaches:

1. Find the action sequence that maximizes the approximated posterior $q(y|x)$;

2. Sample action sequences from $q(y|x)$ and find the one that maximizes $p(y,x)$ (which is proportional to the true posterior). This is a reranking approach similar to (Dyer et al., 2016).
Two approaches:

1. Approximates $\log p(x)$ with lower bound;
Two approaches:

1. Approximates $\log p(x)$ with lower bound;

2. Computes $p(x)$ as $\mathbb{E}_{q(y|x)} \frac{p(x,y)}{q(y|x)}$. This is importance sampling (Dyer et al., 2016) using variational approximation as the proposal distribution (Ghahramani and Beal, 2000).
Comparison to Dyer et al. (2016)

- Dyer et al. (2016) proposes a discriminative and a generative RNNG trained separately;
Comparison to Dyer et al. (2016)

- Dyer et al. (2016) proposes a discriminative and a generative RNNG trained separately;
- We train discriminative and generative RNNGs jointly as a variational autoencoder;
Comparison to Dyer et al. (2016)

- Dyer et al. (2016) proposes a discriminative and a generative RNNG trained separately;
- We train discriminative and generative RNNGs jointly as a variational autoencoder;
- Our model supports both unsupervised and supervised training.
Experiments
Experimental Results: Parsing (English Penn Tree Bank)

<table>
<thead>
<tr>
<th>discriminative parsers</th>
<th></th>
<th>generative parsers</th>
<th></th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhu et al. (2013)</td>
<td>90.4</td>
<td>Petrov and Klein (2007)</td>
<td>90.1</td>
<td>This work: (\text{argmax}_a q(a</td>
</tr>
<tr>
<td>Seq2seq (Vinyals et al., 2015)</td>
<td>88.3</td>
<td>Shindo et al. (2012)</td>
<td>92.4</td>
<td>This work: (\text{argmax}_a p(a, x))</td>
</tr>
<tr>
<td>RNNG (Dyer et al., 2016)</td>
<td>91.7</td>
<td>RNNG (Dyer et al., 2016)</td>
<td>93.3</td>
<td>90.1 + 0.5</td>
</tr>
<tr>
<td>Cross and Huang (2016)</td>
<td>89.9</td>
<td></td>
<td></td>
<td>89.3 + 0.8</td>
</tr>
</tbody>
</table>

**this work: argmax_a q(a|x)**
**Experimental Results: Language Modeling (English Penn Tree Bank)**

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>KN-5</td>
<td>255.2</td>
</tr>
<tr>
<td>LSTM</td>
<td>113.4</td>
</tr>
<tr>
<td>RNNG (Dyer et al., 2016)</td>
<td>102.4</td>
</tr>
<tr>
<td>This work: $a \sim q(a</td>
<td>x)$</td>
</tr>
</tbody>
</table>

**Table 1:** Single-model language modeling results (perplexity).
Summary
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Generative</th>
<th>Discriminative</th>
<th>Our work</th>
</tr>
</thead>
<tbody>
<tr>
<td>models data generation</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>supports both parsing and lan-</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>guage modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consistent training and test</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>objectives</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>supports arbitrary global fea-</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>tures (e.g., sentential fea-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tures)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>supports linear-time recogni-</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>tion algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• We propose a generative model with a discriminative recognition algorithm, which can be used for parsing and language modeling.
• We propose a generative model with a discriminative recognition algorithm, which can be used for parsing and language modeling.
• Future work includes grammar induction with posterior regularization techniques.
We propose a generative model with a discriminative recognition algorithm, which can be used for parsing and language modeling.

Future work includes grammar induction with posterior regularization techniques.

Code available: https://github.com/cheng6076/virnng