Latent Vector Grammars

Kewei Tu
ShanghaiTech University
PCFG with standard syntactic categories

\[
\begin{align*}
S & \rightarrow \text{NP VP} & 1 \\
\text{VP} & \rightarrow V \text{ NP} & 0.2 \\
\text{NP} & \rightarrow \text{DT NP} & 0.5 \\
\text{NP} & \rightarrow \text{NP NP} & 0.3 \\
\ldots \ldots \\
P & \rightarrow \text{He} & 1 \\
P & \rightarrow \text{me} & 1 \\
V & \rightarrow \text{found} & 1 \\
\ldots \ldots \\
\end{align*}
\]

\[
T_1 = \quad T_2 = \\
\begin{array}{c}
S \\
\text{NP} \\
\text{VP} \\
P \\
\text{He} \\
\text{found} \\
\text{me} \\
\end{array} \quad \begin{array}{c}
S \\
\text{NP} \\
\text{VP} \\
P \\
\text{me} \\
\text{found} \\
\text{He} \\
\end{array}
\]

\[P(T_1) = P(T_2)\]
Tree Annotation & Lexicalization

S^ROOT

NP^S  VP^S-BD-Z

P-Z  V-BD-Z  NP^VP-O

P-O

He  found  me

S-found

NP-He  VP-found

P-He  V-found  NP-me

P-me

He  found  me

[Johnson. 1998; Klein et al. 2003]  [Collins. 1997; Charniak. 2000]

Tree Annotation  Lexicalization
Latent Variable Grammars (LVGs)

[Matsuzaki et al. 2005; Petrov et al. 2007]

**Subtype rule**

<table>
<thead>
<tr>
<th>Subtype Rule</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S[0] → NP[0] VP[0]</td>
<td>0.1</td>
</tr>
<tr>
<td>S[1] → NP[0] VP[0]</td>
<td>0.15</td>
</tr>
<tr>
<td>S[0] → NP[1] VP[0]</td>
<td>0.05</td>
</tr>
<tr>
<td>S[1] → NP[1] VP[0]</td>
<td>0.08</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

```
S_0
   / 
 NP_2  VP_0
     |    |
  P_0  V_1  NP_2
    |  |    |
   He found me
```

A subtype parse tree

- Each nonterminal is split into a **finite** number of subtypes
- A **discrete latent variable** is used to indicate the subtype of a nonterminal in a parse tree
- Each subtype rule is associated with a probability
Symbol Embedding

- Important technique in neural NLP: representing discrete symbols with continuous vectors
  - Captures similarity / correlation between symbols
  - More fine-grained representation than symbols
    - Ex: the same symbol in different contexts
  - Flexible representation of symbolic operations
  - Informed smoothing in prediction
  - Makes end-to-end learning possible

- Idea: vectorize weighted CFG
- Previous work: Compositional Vector Grammars (CVGs) [Socher et al., 2011 & 2013]
  - Problem: exact inference is intractable
Latent Vector Grammars (LVeGs)

Each nonterminal has a continuous vector space representing an infinite number of subtypes.

A continuous latent vector is used to indicate the subtype of a nonterminal in a parse tree.

A continuous weight density function specifies the weight of any subtype rule.

Subtype rule | Weight Density
--- | ---
S[0.05, 0.2] → NP[0.4, 1.7] VP[3.4, 0.9] 1.5
NP[0.4, 1.7] → P[0.3, 2.1] 0.7

A subtype parse tree

- Each nonterminal has a continuous vector space representing an infinite number of subtypes.
- A continuous latent vector is used to indicate the subtype of a nonterminal in a parse tree.
- A continuous weight density function specifies the weight of any subtype rule.
<table>
<thead>
<tr>
<th>LVGs (Latent Variable Grammars)</th>
<th>vs</th>
<th>LVeGs (Latent Vector Grammars)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A small fixed number of subtypes</td>
<td></td>
<td>Unlimited number of subtypes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Potentially more representational power</td>
</tr>
<tr>
<td>Number of subtypes is manually specified or learned by split-merge</td>
<td></td>
<td>Can control the level of subtype granularity by weight function smoothness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ More elegant</td>
</tr>
<tr>
<td>No indication of similarity between subtypes</td>
<td></td>
<td>Similarity between subtypes are modeled by distance between vectors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Informed smoothing</td>
</tr>
</tbody>
</table>
Parsing with LV(e)Gs

What we have
A distribution over subtype parse trees

What we want
The unannotated parse tree with the highest score

Intractable!
Parsing with LV(e)Gs

Max-Rule-Prod [Petrov et al., 2007]

\[ T^*_q = \arg\max_{T \in G(w)} \prod_{e \in T} q(e) \]

\( q(e) \) is the expected count of an anchored rule \( e = \langle r, i, j, k \rangle \), i.e., the total weight of all the subtype parse trees that use rule \( r \) at spans \( \langle i, j, k \rangle \).

\( q(e) \) can be computed by inside and outside scores:

\[ q(A \rightarrow BC, i, k, j) = \frac{s(A \rightarrow BC, i, k, j)}{s_I(S, 1, n)} \]

\( s_I(S, 1, n) \) is computable from inside and outside scores.

Inside score

Computable from inside and outside scores
Parsing with LVeGs

Recursive computation of inside (and outside) scores

\[ s_i^A(a, i, j) = \sum_{A \rightarrow BC \in R} \int \int W_{A \rightarrow BC}(a, b, c) \times s_i^B(b, i, k) \times s_i^C(c, k + 1, j) \, dbdc \]

• Key observation: Gaussian mixtures are closed under integral, product, and summation

\[ W_r(r) = \sum_{i=1}^{K_r} \rho_{r, i} \mathcal{N}(r | \mu_{r, i}, \Sigma_{r, i}) \]

• If the weight density functions are Gaussian mixtures, then the inside and outside scores have closed-form solutions and can be computed with DP

Gaussian Mixture LVeGs
Pruning

- **Component Pruning**
  - Limit the number of Gaussian components of inside and outside score functions

- **Constituent Pruning**
  - Prune less probable syntactic categories for every span
  - Consider only the constituents in k-best parses
Learning with GM-LVeGs

Discriminative learning

$$\mathcal{L}(\Theta) = -\log \prod_{i=1}^{m} P(T_i|w_i; \Theta)$$

Optimizing using Adam

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta_r} = \sum_{i=1}^{m} \int \left( \frac{\partial W_r(r)}{\partial \Theta_r} \times \frac{\mathbb{E}_{P(t|w_i)}[f_r(t)] - \mathbb{E}_{P(t|T_i)}[f_r(t)]}{W_r(r)} \right) dr$$

Closed form solution exists with GM-LVeGs
# Experiments on POS Tagging

<table>
<thead>
<tr>
<th>Model</th>
<th>WSJ</th>
<th>English</th>
<th>French</th>
<th>German</th>
<th>Russian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>S</td>
<td>T</td>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>LVG-D-16</td>
<td>96.62</td>
<td>48.74</td>
<td>92.31</td>
<td>52.67</td>
<td>93.75</td>
</tr>
<tr>
<td>LVG-G-16</td>
<td>96.78</td>
<td>50.88</td>
<td>93.30</td>
<td>57.54</td>
<td>94.52</td>
</tr>
<tr>
<td>GM-LVeG-D</td>
<td>96.99</td>
<td>53.10</td>
<td>93.66</td>
<td>59.46</td>
<td>94.73</td>
</tr>
<tr>
<td>GM-LVeG-S</td>
<td>97.00</td>
<td>53.11</td>
<td>93.55</td>
<td>58.11</td>
<td>94.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Spanish</th>
<th>Indonesian</th>
<th>Finnish</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>S</td>
<td>T</td>
<td>S</td>
</tr>
<tr>
<td>LVG-D-16</td>
<td>92.47</td>
<td>24.82</td>
<td>89.27</td>
<td>20.29</td>
</tr>
<tr>
<td>LVG-G-16</td>
<td>93.21</td>
<td>27.37</td>
<td>90.09</td>
<td>21.19</td>
</tr>
<tr>
<td>GM-LVeG-D</td>
<td>93.76</td>
<td>32.48</td>
<td>90.24</td>
<td>21.72</td>
</tr>
<tr>
<td>GM-LVeG-S</td>
<td>93.52</td>
<td>30.66</td>
<td>90.12</td>
<td>21.72</td>
</tr>
</tbody>
</table>

### Experiments on Parsing

<table>
<thead>
<tr>
<th>Model</th>
<th>dev (all)</th>
<th>test ≤ 40</th>
<th>test (all)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F1</td>
<td>EX</td>
</tr>
<tr>
<td>LVG-G-16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVG-D-16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-Scale</td>
<td>89.70</td>
<td>39.60</td>
<td></td>
</tr>
<tr>
<td>Berkeley Parser</td>
<td>90.60</td>
<td>39.10</td>
<td></td>
</tr>
<tr>
<td>CVG (SU-RNN)</td>
<td>91.20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>91.24</strong></td>
<td><strong>91.38</strong></td>
<td><strong>41.51</strong></td>
</tr>
</tbody>
</table>

- LVG-G/D (Petrov and Klein 2008a)
- Multi-Scale Grammars (Petrov and Klein 2008b)
- Berkeley Parser (Petrov and Klein 2007)
- CVG (SU-RNN) (Socher et al., 2013)

(no contextual info used for fair comparison)
Summary

- Nonterminal subtyping is important in modern constituency parsing
- Latent Vector Grammars (LVeGs)
  - Each nonterminal has a continuous vector space representing an infinite number of subtypes
- Gaussian Mixture LVeGs (GM-LVeGs)
  - Rule weight functions formulated with Gaussian mixtures
  - Efficient estimation in learning and inference
  - Competitive results on POS tagging and constituency parsing
Thank you!

Q&A