Deep Residual Output Layers for Neural Language Generation

Nikolaos Pappas, James Henderson

June 13, 2019
Neural language generation

Probability distribution at time $t$ given context vector $h_t \in \mathbb{R}^d$, weights $W \in \mathbb{R}^{d \times |V|}$ and bias $b \in \mathbb{R}^{|V|}$:

$$p(y_t|y_{1:t-1}) \propto \exp(W^T h_t + b)$$
Neural language generation

Probability distribution at time $t$ given context vector $h_t \in \mathbb{R}^d$, weights $W \in \mathbb{R}^{d \times |V|}$ and bias $b \in \mathbb{R}^{|V|}$:

$$p(y_t|y_{1:t-1}) \propto \exp(W^T h_t + b)$$

- Output layer parameterisation depends on the vocabulary size $|V|$
  - Sample inefficient
Neural language generation

Probability distribution at time $t$ given context vector $h_t \in \mathbb{R}^d$, weights $W \in \mathbb{R}^{d \times |V|}$ and bias $b \in \mathbb{R}^{|V|}$:

$$p(y_t | y_1^{t-1}) \propto \exp(W^T h_t + b)$$

- Output layer parameterisation depends on the vocabulary size $|V|$
  - **Sample inefficient**

- Output layer power depends on hidden dim or rank $d$: “softmax bottleneck”
  - **High overhead and prone to overfitting**
Previous work

Probability distribution at time $t$ given context vector $h_t \in \mathbb{R}^d$, weights $W \in \mathbb{R}^{d \times |\mathcal{V}|}$ and bias $b \in \mathbb{R}^{|\mathcal{V}|}$:

$$p(y_t | y_{1:t-1}) \propto \exp(W^T h_t + b)$$

- Output layer parameterisation no longer depends on the vocabulary size $|\mathcal{V}|$ \hspace{1cm} (1)  
  → More sample efficient
- Output layer power still depends on hidden dim or rank $d$: “softmax bottleneck” \hspace{1cm} (2)  
  → High overhead and prone to overfitting

Output similarity structure learning methods help with (1) but not yet with (2).
Previous work

Output structure learning factorization of probability distribution given word embedding $E \in \mathbb{R}^{|V| \times d}$:

$$p(y_t | y_{1:t-1}) \propto g_{out}(E, V)g_{in}(E, y_{1:t-1}) + b$$

- Shallow label encoder networks such as weight tying [PW17], bilinear mapping [G18], and dual nonlinear mapping [P18]
Our contributions

Output structure learning factorization of probability distribution given word embedding $E \in \mathbb{R}^{|V| \times d}$:

$$p(y_t|y_{1}^{t-1}) \propto g_{out}(E, \mathcal{V})g_{in}(E, y_{1}^{t-1}) + b$$

- Generalize previous output similarity structure learning methods
  → **More sample efficient**
Our contributions

Output structure learning factorization of probability distribution given word embedding $E \in \mathbb{R}^{|\mathcal{V}| \times d}$:

$$p(y_t | y_{1}^{t-1}) \propto g_{out}(E, \mathcal{V})g_{in}(E, y_{1}^{t-1}) + b$$

- Generalize previous output similarity structure learning methods
  → More sample efficient
- Propose a deep output label encoder network with dropout between layers
  → Avoids overfitting
Our contributions

Output structure learning factorization of probability distribution given word embedding $E \in \mathbb{R}^{|V| \times d}$:

$$p(y_t|y_{1:t-1}) \propto g_{out}(E, V)g_{in}(E, y_{1:t-1}) + b$$

- Generalize previous output similarity structure learning methods
  → **More sample efficient**
- Propose a deep output label encoder network with dropout between layers
  → **Avoids overfitting**
- Increase output layer power with representation depth instead of rank $d$
  → **Low overhead**
Label Encoder Network

- Shares parameters across output labels with \( k \) nonlinear projections

\[
E^{(k)} = f^{(k)}_{out}(E^{(k-1)})
\]
Label Encoder Network

- Shares parameters across output labels with \( k \) nonlinear projections
  \[ E^{(k)} = f_{out}^{(k)}(E^{(k-1)}) \]

- Preserves information across layers with residual connections
  \[ E^{(k)} = f_{out}^{(k)}(E^{(k-1)}) + E^{(k-1)} + E \]
Label Encoder Network

- Shares parameters across output labels with \( k \) nonlinear projections

\[
E^{(k)} = f_{out}^{(k)}(E^{(k-1)})
\]

- Preserves information across layers with residual connections

\[
E^{(k)} = f_{out}^{(k)}(E^{(k-1)}) + E^{(k-1)} + E
\]

- Avoids overfitting with standard or variational dropout for each layer \( i = 1, \ldots, k \)

\[
f_{out}^{(i)}(E^{(i-1)}) = \delta(f_{out}^{(i)}(E^{(i-1)})) \odot f_{out}^{(i)}(E^{(i-1)})
\]
Results

- Improve competitive architectures without increasing their dim or rank

<table>
<thead>
<tr>
<th>Language modeling</th>
<th>ppl</th>
<th>sec/ep</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWD-LSTM [M18]</td>
<td>65.8</td>
<td>89 (1.0×)</td>
</tr>
<tr>
<td>AWD-LSTM-DRILL</td>
<td>61.9</td>
<td>106 (1.2×)</td>
</tr>
<tr>
<td>AWD-LSTM-MoS [Y18]</td>
<td>61.4</td>
<td>862 (9.7×)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machine translation</th>
<th>bleu</th>
<th>min/ep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer [V17]</td>
<td>27.3</td>
<td>111 (1.0×)</td>
</tr>
<tr>
<td>Transformer-DRILL</td>
<td>28.1</td>
<td>189 (1.7×)</td>
</tr>
<tr>
<td>Transformer (big) [V17]</td>
<td>28.4</td>
<td>779 (7.0×)</td>
</tr>
</tbody>
</table>

- Better transfer across low-resource output labels

![Graph comparing weight tying, bilinear map, and dual nonlinear map with DRILL]
Talk to us at Poster #104 in Pacific Ballroom.

Thank you!

http://github.com/idiap/drill