Sequence-to-Sequence Learning
as Beam-Search Optimization

Sam Wiseman and Alexander M. Rush
Seq2Seq as a General-purpose NLP/Text Generation Tool

- Machine Translation
- Question Answering
- Conversation
- Parsing
- Sentence Compression
- Summarization
- Caption Generation
- Video-to-Text
- Grammar Correction
Despite its tremendous success, there are some potential issues with standard Seq2Seq [Ranzato et al. 2016; Bengio et al. 2015]:

(1) Train/Test mismatch

(2) Seq2Seq models next-words, rather than whole sequences

**Goal of the talk:** describe a simple variant of Seq2Seq — and corresponding beam-search training scheme — to address these issues.
- **Encoder RNN (red)** encodes source into a representation $x$

- **Decoder RNN (blue)** generates translation word-by-word
Probability of generating $t$'th word:

$$p(w_t|w_1, \ldots, w_{t-1}, x; \theta) = \text{softmax}(W_{out} h_{t-1} + b_{out})$$
Train Objective: Given source-target pairs \((x, y_{1:T})\), minimize NLL of each word independently, conditioned on gold history \(y_{1:t-1}\)

\[
\text{NLL}(\theta) = - \sum_t \ln p(w_t = y_t | y_{1:t-1}, x; \theta)
\]

Test Objective: Structured prediction

\[
\hat{y}_{1:T} = \arg \max_{w_{1:T}} \sum_t \ln p(w_t | w_{1:t-1}, x; \theta)
\]

- Typical to approximate the \(\arg \max\) with beam-search
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

$$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

$$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

\[
s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)}|\mathbf{x}) + \ln p(w_t = w|\hat{y}_{1:t-1}^{(k)}, \mathbf{x})\]

- Update beam:

\[
\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max } s(w_t, \hat{y}_{1:t-1}^{(k)})
\]
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:
  \[
  s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x})
  \]

- Update beam:
  \[
  \hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max } \max_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})
  \]
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:
  \[
  s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)}|\mathbf{x}) + \ln p(w_t = w|\hat{y}_{1:t-1}^{(k)}, \mathbf{x})
  \]

- Update beam:
  \[
  \hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})
  \]
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

  $$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

  $$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max }_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

$$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

$$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

$$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)}|\mathbf{x}) + \ln p(w_t = w|\hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

$$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

  $$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)}|\mathbf{x}) + \ln p(w_t = w|\hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

  $$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max } \max_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

  $$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

  $$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

$$s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x})$$

- Update beam:

$$\hat{y}_{1:t}^{(1:K)} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}_{1:t-1}^{(k)})$$
For $t = 1 \ldots T$:

- For all $k$ and for all possible output words $w$:

$$s(w_t = w, \hat{y}^{(k)}_{1:t-1}) \leftarrow \ln p(\hat{y}^{(k)}_{1:t-1} | \mathbf{x}) + \ln p(w_t = w | \hat{y}^{(k)}_{1:t-1}, \mathbf{x})$$

- Update beam:

$$\hat{y}^{(1:K)}_{1:t} \leftarrow \text{K-arg max}_{w_{1:t}} s(w_t, \hat{y}^{(k)}_{1:t-1})$$
Issue #1: Train/Test Mismatch (cf., Ranzato et al. [2016])

\[
\text{NLL}(\theta) = - \sum_t \ln p(w_t = y_t | y_{1:t-1}, x; \theta)
\]

(a) Training conditions on true history (“Exposure Bias”)

(b) Train with word-level NLL, but evaluate with BLEU-like metrics

Idea #1: Train with beam-search

- Use a loss that incorporates (sub)sequence-level costs
Seq2Seq Issues Revisited

**Issue #1: Train/Test Mismatch** (cf., Ranzato et al. [2016])

\[
\text{NLL}(\theta) = - \sum_t \ln p(w_t = y_t | y_{1:t-1}, x; \theta)
\]

(a) Training conditions on *true* history ("Exposure Bias")
(b) Train with word-level NLL, but evaluate with BLEU-like metrics

**Idea #1:** Train with beam-search

- Use a loss that incorporates (sub)sequence-level costs
**Issue #1: Train/Test Mismatch** (cf., Ranzato et al. [2016])

\[
\text{NLL}(\theta) = - \sum_t \ln p(w_t = y_t | y_{1:t-1}, x; \theta)
\]

(a) Training conditions on *true* history (“Exposure Bias”)

(b) Train with word-level NLL, but evaluate with BLEU-like metrics

**Idea #1:** Train with beam-search

- Use a loss that incorporates (sub)sequence-level costs
**Issue #1: Train/Test Mismatch** (cf., Ranzato et al. [2016])

\[ \text{NLL}(\theta) = - \sum_t \ln p(w_t = y_t | y_{1:t-1}, x; \theta) \]

(a) Training conditions on *true* history ("Exposure Bias")

(b) Train with word-level NLL, but evaluate with BLEU-like metrics

**Idea #1:** Train with beam-search

- Use a loss that incorporates (sub)sequence-level costs
Idea #1: Train with Beam Search

Replace NLL with loss that penalizes search-error:

\[
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
\]

- \(y_{1:t}\) is the gold prefix; \(\hat{y}_{1:t}^{(K)}\) is the \(K\)'th prefix on the beam
- \(s(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})\) is the score of history \((\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})\)
- \(\Delta(\hat{y}_{1:t}^{(K)})\) allows us to scale loss by badness of predicting \(\hat{y}_{1:t}^{(K)}\)
Idea #1: Train with Beam Search

Replace NLL with loss that penalizes search-error:

$$
L(\theta) = \sum_t \Delta(\hat{y}^{(K)}_{1:t}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
$$

- $y_{1:t}$ is the gold prefix; $\hat{y}^{(K)}_{1:t}$ is the $K$'th prefix on the beam
- $s(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})$ is the score of history $(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})$
- $\Delta(\hat{y}^{(K)}_{1:t})$ allows us to scale loss by badness of predicting $\hat{y}^{(K)}_{1:t}$
Idea #1: Train with Beam Search

Replace NLL with loss that penalizes search-error:

$$
L(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
$$

- $y_{1:t}$ is the gold prefix; $\hat{y}_{1:t}^{(K)}$ is the $K$’th prefix on the beam
- $s(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})$ is the score of history $(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})$
- $\Delta(\hat{y}_{1:t}^{(K)})$ allows us to scale loss by badness of predicting $\hat{y}_{1:t}^{(K)}$
Replace NLL with loss that penalizes search-error:

$$
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
$$

- $y_{1:t}$ is the gold prefix; $\hat{y}_{1:t}^{(K)}$ is the $K$’th prefix on the beam
- $s(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})$ is the score of history $(\hat{y}_t^{(k)}, \hat{y}_{1:t-1}^{(k)})$
- $\Delta(\hat{y}_{1:t}^{(K)})$ allows us to scale loss by badness of predicting $\hat{y}_{1:t}^{(K)}$
Issue #2: Seq2Seq models next-word probabilities:

\[ s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x}) \]

(a) Sequence score is sum of locally normalized word-scores; gives rise to “Label Bias” [Lafferty et al. 2001]

(b) What if we want to train with sequence-level constraints?

Idea #2: Don’t locally normalize
Issue #2: Seq2Seq models next-word probabilities:

\[ s(w_t = w, \hat{y}_{1:t-1}) \leftarrow \ln p(\hat{y}_{1:t-1} | \boldsymbol{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}, \boldsymbol{x}) \]

(a) Sequence score is sum of locally normalized word-scores; gives rise to “Label Bias” [Lafferty et al. 2001]

(b) What if we want to train with sequence-level constraints?

Idea #2: Don’t locally normalize
Issue #2: Seq2Seq models next-word probabilities:

\[ s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x}) \]

(a) Sequence score is sum of locally normalized word-scores; gives rise to “Label Bias” [Lafferty et al. 2001]

(b) What if we want to train with sequence-level constraints?

Idea #2: Don’t locally normalize
Issue #2: Seq2Seq models next-word probabilities:

\[ s(w_t = w, \hat{y}_{1:t-1}^{(k)}) \leftarrow \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{x}) + \ln p(w_t = w | \hat{y}_{1:t-1}^{(k)}, \mathbf{x}) \]

(a) Sequence score is sum of locally normalized word-scores; gives rise to “Label Bias” [Lafferty et al. 2001]

(b) What if we want to train with sequence-level constraints?

Idea #2: Don’t locally normalize
Idea #2: Don’t locally normalize

\[ h^{(k)}_1 \rightarrow h^{(k)}_2 \rightarrow h^{(k)}_3 = \text{RNN}(y^{(k)}_3, h^{(k)}_2) \]

\[ y^{(k)}_1 \rightarrow y^{(k)}_2 \rightarrow y^{(k)}_3 \]

\[ s(w, \hat{y}_{1:t-1}^{(k)}) = \ln p(\hat{y}_{1:t-1}^{(k)}|\mathbf{x}) + \ln \text{softmax}(W_{out} h^{(k)}_{t-1} + b_{out}) \]
Idea #2: Don’t locally normalize

\[ h^{(k)}_1 h^{(k)}_2 h^{(k)}_3 = \text{RNN}(y^{(k)}_3, h^{(k)}_2) \]

\[
s(w, \hat{y}^{(k)}_{1:t-1}) = \ln p(\hat{y}^{(k)}_{1:t-1} | x) + \ln \text{softmax}(W_{out} h^{(k)}_{t-1} + b_{out})
\]
\[
= W_{out} h^{(k)}_{t-1} + b_{out}
\]
Idea #2: Don’t locally normalize

\[s(w, \hat{y}_{1:t-1}^{(k)}) = \ln p(\hat{y}_{1:t-1}^{(k)} | \mathbf{h}_{t-1}) + \ln \text{softmax}(\mathbf{W}_{out} \mathbf{h}_{t-1}^{(k)} + \mathbf{b}_{out})\]

\[= \mathbf{W}_{out} \mathbf{h}_{t-1}^{(k)} + \mathbf{b}_{out}\]

- Can set \(s(w, \hat{y}_{1:t-1}^{(k)}) = -\infty\) if \((w, \hat{y}_{1:t-1}^{(k)})\) violates a hard constraint
Computing Gradients of the Loss \((K = 3)\)

\[
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
\]

- **Color Gold**: target sequence \(y\)
- **Color Gray**: violating sequence \(\hat{y}^{(K)}\)
Computing Gradients of the Loss \((K = 3)\)

\[
\mathcal{L}(\theta) = \sum_{t} \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
\]

- **Color Gold**: target sequence \(y\)
- **Color Gray**: violating sequence \(\hat{y}^{(K)}\)
Computing Gradients of the Loss \((K = 3)\)

\[
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
\]

- **Color Gold**: target sequence \(y\)
- **Color Gray**: violating sequence \(\hat{y}^{(K)}\)
Computing Gradients of the Loss ($K = 3$)

\[
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_1^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
\]

- **Color Gold**: target sequence $y$
- **Color Gray**: violating sequence $\hat{y}^{(K)}$
Computing Gradients of the Loss ($K = 3$)

\[ \mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right] \]

- Need to BPTT for both $y_{1:t}$ and $\hat{y}_{1:t}^{(K)}$, which is $O(T)$
- Worst case: violation at each $t$ gives $O(T^2)$ backward pass
Computing Gradients of the Loss \((K = 3)\)

\[
L(\theta) = \sum_t \Delta(\hat{y}^{(K)}_{1:t}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}^{(K)}_t, \hat{y}^{(K)}_{1:t-1}) \right]
\]

- Need to BPTT for both \(y_{1:t}\) and \(\hat{y}^{(K)}_{1:t}\), which is \(O(T)\)

- Worst case: violation at each \(t\) gives \(O(T^2)\) backward pass

- **Idea:** use LaSO [Daumé III and Marcu 2005] beam-update
Computing Gradients of the Loss \((K = 3)\)

\[
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]
\]

- Need to BPTT for both \(y_{1:t}\) and \(\hat{y}_{1:t}^{(K)}\), which is \(O(T)\)
- Worst case: violation at each \(t\) gives \(O(T^2)\) backward pass
- **Idea:** use LaSO [Daumé III and Marcu 2005] beam-update
Computing Gradients of the Loss ($K = 3$)

\[ \mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right] \]

**LaSO** [Daumé III and Marcu 2005]:

- If no margin violation at $t-1$, update beam as usual

- Otherwise, update beam with sequences prefixed by $y_{1:t-1}$
Computing Gradients of the Loss \((K = 3)\)

\[
\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}^{(K)}_{1:t}) \left[ 1 - s(y_t, y_{1:t-1}) + s(\hat{y}^{(K)}_t, \hat{y}^{(K)}_{1:t-1}) \right]
\]

**LaSO** [Daumé III and Marcu 2005]:

- If no margin violation at \(t - 1\), update beam as usual
- Otherwise, update beam with sequences prefixed by \(y_{1:t-1}\)
Margin gradients are sparse, only violating sequences get updates.

Backprop only requires $2x$ time as standard methods.
(Recent) Related Work and Discussion

Recent approaches to Exposure Bias, Label Bias:
- Data as Demonstrator, Scheduled Sampling [Bengio et al. 2015]
- Globally Normalized Transition-Based Networks

RL-based approaches
- MIXER [Ranzato et al. 2016]
- Actor-Critic

Training with beam-search attempts to offer similar benefits
- Uses fact that we typically have gold prefixes in supervised text-generation to avoid RL
Experiments

Experiments run on three Seq2Seq baseline tasks:

- Word Ordering, Dependency Parsing, Machine Translation

We compare with Yoon Kim’s implementation\(^1\) of the Seq2Seq architecture of ?.

- Uses LSTM encoders and decoders, attention, input feeding
- All models trained with Adagrad [Duchi et al. 2011]
- Pre-trained with NLL; \(K\) increased gradually
- “BSO” uses unconstrained search; “ConBSO” uses constraints

\(^1\)https://github.com/harvardnlp/seq2seq-attn
Word Ordering Experiments

<table>
<thead>
<tr>
<th></th>
<th>Word Ordering (BLEU)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{te} = 1$</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>25.2</td>
</tr>
<tr>
<td>BSO</td>
<td>28.0</td>
</tr>
<tr>
<td>ConBSO</td>
<td><strong>28.6</strong></td>
</tr>
</tbody>
</table>

- Map shuffled sentence to correctly ordered sentence
- Same setup as Liu et al. [2015]
- BSO models trained with beam of size 6
## Word Ordering Experiments

<table>
<thead>
<tr>
<th></th>
<th>Word Ordering (BLEU)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{te} = 1$</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>25.2</td>
</tr>
<tr>
<td>BSO</td>
<td>28.0</td>
</tr>
<tr>
<td>ConBSO</td>
<td><strong>28.6</strong></td>
</tr>
</tbody>
</table>

- Map shuffled sentence to correctly ordered sentence
- Same setup as Liu et al. [2015]
- BSO models trained with beam of size 6
### Word Ordering Experiments

<table>
<thead>
<tr>
<th></th>
<th>$K_{te} = 1$</th>
<th>$K_{te} = 5$</th>
<th>$K_{te} = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>25.2</td>
<td>29.8</td>
<td>31.0</td>
</tr>
<tr>
<td>BSO</td>
<td>28.0</td>
<td>33.2</td>
<td>34.3</td>
</tr>
<tr>
<td>ConBSO</td>
<td><strong>28.6</strong></td>
<td><strong>34.3</strong></td>
<td><strong>34.5</strong></td>
</tr>
</tbody>
</table>

- Map shuffled sentence to correctly ordered sentence
- Same setup as Liu et al. [2015]
- BSO models trained with beam of size 6
Dependency Parsing Experiments

Source: Ms. Haag plays Elianti.

Target: Ms. Haag @L_NN plays @L_NSUBJ Elianti @R_DOBJ . @R_PUNCT

<table>
<thead>
<tr>
<th>Dependency Parsing (UAS/LAS)</th>
<th>$K_{te} = 1$</th>
<th>$K_{te} = 5$</th>
<th>$K_{te} = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>87.33/82.26</td>
<td>88.53/84.16</td>
<td>88.66/84.33</td>
</tr>
<tr>
<td>BSO</td>
<td>86.91/82.11</td>
<td>91.00/87.18</td>
<td>91.17/87.41</td>
</tr>
<tr>
<td>ConBSO</td>
<td>85.11/79.32</td>
<td>91.25/86.92</td>
<td>91.57/87.26</td>
</tr>
</tbody>
</table>

- BSO models trained with beam of size 6
- Same setup and evaluation as Chen and Manning [2014]
- Certainly not SOA, but reasonable for word-only, left-to-right model
## Machine Translation: Impact of Non-0/1 \( \Delta \)

<table>
<thead>
<tr>
<th>( \Delta(\hat{y}_{1:t}^{(k)}) = 1 { \text{margin violation} } )</th>
<th>( \Delta(\hat{y}<em>{1:t}^{(k)}) = 1 - \text{SentBLEU}(\hat{y}</em>{r+1:t}^{(K)}, y_{r+1:t}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Translation (BLEU) ( K_{te} = 1 )</td>
<td>25.73</td>
</tr>
<tr>
<td>( K_{te} = 5 )</td>
<td>28.21</td>
</tr>
<tr>
<td>( K_{te} = 10 )</td>
<td>27.43</td>
</tr>
</tbody>
</table>

- IWSLT 2014, DE-EN, development set
- BSO models trained with beam of size 6
- Nothing to write home about, but nice that we can tune to metrics
### Machine Translation Experiments

<table>
<thead>
<tr>
<th></th>
<th>Machine Translation (BLEU)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{te} = 1$</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>22.53</td>
</tr>
<tr>
<td>BSO</td>
<td>23.83</td>
</tr>
<tr>
<td>NLL</td>
<td>17.74</td>
</tr>
<tr>
<td>DAD [?]</td>
<td>20.12</td>
</tr>
</tbody>
</table>

- **IWSLT 2014, DE-EN**
- BSO models trained with beam of size 6
- $\Delta(\hat{y}^{(k)}_{1:t}) = 1 - \text{SentBLEU}(\hat{y}^{(K)}_{r+1:t}, y_{r+1:t})$
- Results in bottom sub-table from Ranzato et al. [2016]
- Note similar improvements to MIXER
# Machine Translation Experiments

<table>
<thead>
<tr>
<th>Model</th>
<th>Machine Translation (BLEU)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{te} = 1$</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>22.53</td>
</tr>
<tr>
<td>BSO</td>
<td><strong>23.83</strong></td>
</tr>
<tr>
<td>NLL</td>
<td>17.74</td>
</tr>
<tr>
<td>DAD [?]</td>
<td>20.12</td>
</tr>
</tbody>
</table>

- **IWSLT 2014, DE-EN**
- BSO models trained with beam of size 6
- $\Delta(\hat{y}^{(k)}_{1:t}) = 1 - \text{SentBLEU}(\hat{y}^{(K)}_{r+1:t}, y_{r+1:t})$
- Results in bottom sub-table from Ranzato et al. [2016]
- Note similar improvements to MIXER
Conclusion

Introduced a variant of Seq2Seq and training procedure that:

- Attempts to mitigate Label Bias and Exposure Bias
- Allows tuning to test-time metrics
- Allows training with hard constraints
- Doesn’t require RL

N.B. Backprop through search is a thing now/again:

- One piece of the CCG parsing approach of Lee et al. (2016), an EMNLP 2016 Best Paper!
Thanks!
## Training with Different Beam Sizes

<table>
<thead>
<tr>
<th></th>
<th>Word Ordering Beam Size (BLEU)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{te} = 1$</td>
<td>$K_{te} = 5$</td>
<td>$K_{te} = 10$</td>
<td></td>
</tr>
<tr>
<td>$K_{tr} = 2$</td>
<td>30.59</td>
<td>31.23</td>
<td>30.26</td>
<td></td>
</tr>
<tr>
<td>$K_{tr} = 6$</td>
<td>28.20</td>
<td>34.22</td>
<td>34.67</td>
<td></td>
</tr>
<tr>
<td>$K_{tr} = 11$</td>
<td>26.88</td>
<td>34.42</td>
<td>34.88</td>
<td></td>
</tr>
</tbody>
</table>

- ConBSO model, development set results
1: procedure BSO(\(x, K_{tr}, succ\))
2:  Init empty storage \(\hat{y}_{1:T}\) and \(\hat{h}_{1:T}\); init \(S_1\)
3:  \(r \leftarrow 0\); violations \(\leftarrow \{0\}\)
4:  for \(t = 1, \ldots, T\) do  \(\triangleright\) Forward
5:  \(K = K_{tr}\) if \(t \neq T\) else arg max \(f(\hat{y}^{(k)}_{t}, \hat{h}_{t-1}^{(k)})\)
6:  \(k : \hat{y}^{(k)}_{1:t} \neq y_{1:t}\)
7:  if \(f(y_t, h_{t-1}) < f(\hat{y}^{(K)}_{t}, \hat{h}_{t-1}^{(K)}) + 1\) then
8:    \(\hat{h}_{r:t-1} \leftarrow \hat{h}_{r:t-1}^{(K)}\)
9:    \(\hat{y}_{r+1:t} \leftarrow \hat{y}_{r+1:t}^{(K)}\)
10:   Add \(t\) to violations; \(r \leftarrow t\)
11:  else
12:    \(S_{t+1} \leftarrow \text{topK}(\text{succ}(y_{1:t}))\)
13:  end if
14:  grad \(h_T \leftarrow 0\); grad \(\hat{h}_T \leftarrow 0\)
15:  for \(t = T - 1, \ldots, 1\) do  \(\triangleright\) Backward
16:    grad \(h_t \leftarrow \text{BRNN}(\nabla h_t L_{t+1}, \text{grad} h_{t+1})\)
17:    grad \(\hat{h}_t \leftarrow \text{BRNN}(\nabla \hat{h}_t L_{t+1}, \text{grad} \hat{h}_{t+1})\)
18:  if \(t - 1 \in \text{violations}\) then
19:    grad \(h_t \leftarrow \text{grad} h_t + \text{grad} \hat{h}_t\)
20:    grad \(\hat{h}_t \leftarrow 0\)
Margin gradients are sparse, only violating sequences get updates.

Backprop only requires 2x time as standard methods.


Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with


