Sequence-to-Sequence Learning as Beam-Search Optimization

Sam Wiseman and Alexander M. Rush

SEASLogo.png

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 **?**

Room for Improvement?

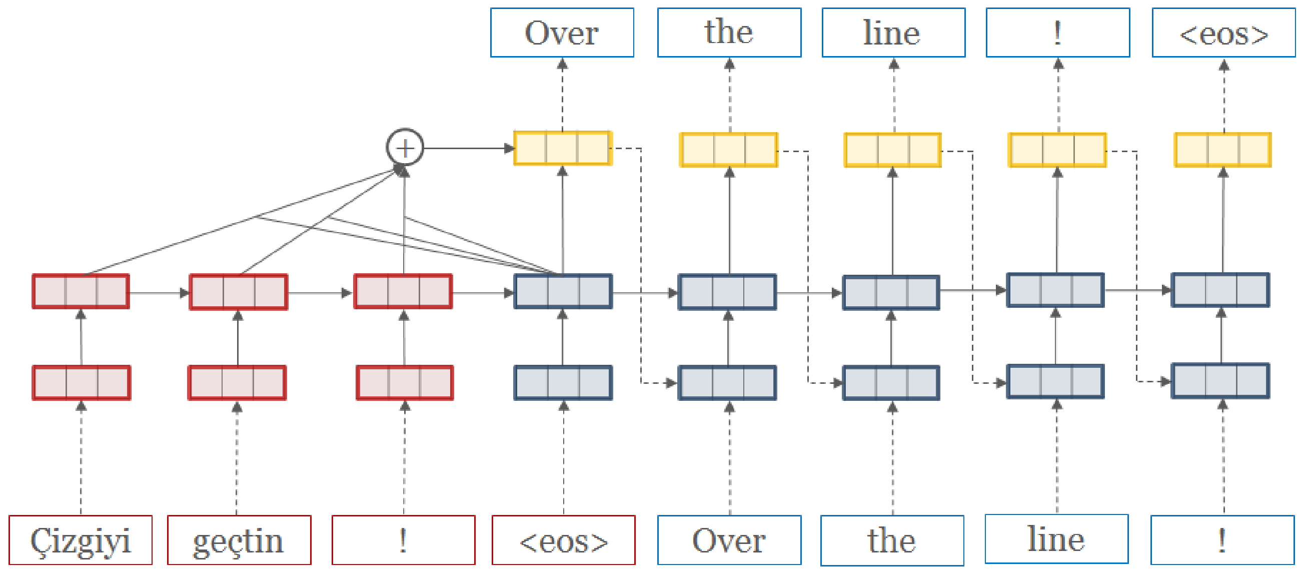
Despite its tremendous success, there are some potential issues with standard Seq2Seq [**??**]:

(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.

Review: Sequence-to-sequence (Seq2Seq) Models



- Encoder RNN (red) encodes source into a representation x
- Decoder RNN (blue) generates translation word-by-word

Review: Seq2Seq Generation Details

- Probability of generating t'th word:

$$p(w_t | w_1, \ldots, w_{t-1}, \boldsymbol{x}; \theta) = \text{softmax}(\mathbf{W}_{out}\,\mathbf{h}_{t-1} + \mathbf{b}_{out})$$

Review: Train and Test

Train Objective: Given source-target pairs $(x, y_{1:T})$, minimize NLL of each word independently, conditioned on *gold* history $y_{1:t-1}$

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

Test Objective: Structured prediction

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

- Typical to approximate the $\arg\max$ with beam-search

Review: Beam Search at Test Time ($K = 3$)

a

the

red

For $t = 1 \dots T$:

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

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)

a

the

red

For $t = 1 \dots T$:

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

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)

a
the
red

For $t = 1 \ldots T$:

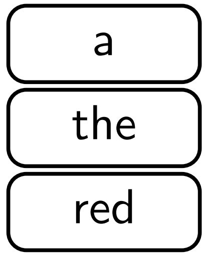
- For all k and for all possible output words w:

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)



For $t = 1 \dots T$:

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

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)

a

the

red

For $t = 1 \dots T$:

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

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

- Update beam:

$$\color{red}{\hat{y}^{(1:K)}_{1:t}} \leftarrow \underset{w_{1:t}}{\text{K-arg max}}\ s(w_t, \hat{y}^{(k)}_{1:t-1})$$

Review: Beam Search at Test Time ($K = 3$)

a

the

red

For $t = 1 \dots T$:

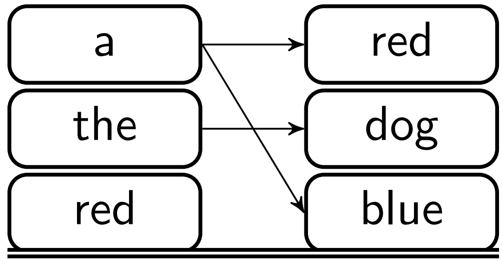
- For all k and for all possible output words w:

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)



For $t = 1 \ldots T$:

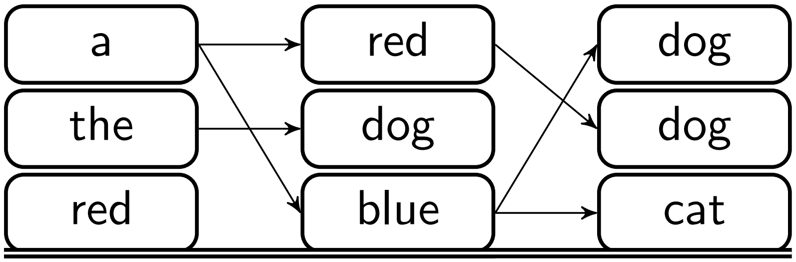
- For all k and for all possible output words w:

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)



For $t = 1 \dots T$:

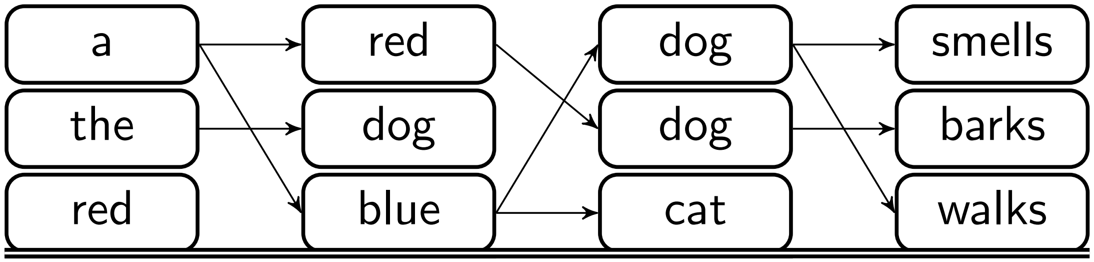
- For all k and for all possible output words w:

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)



For $t = 1 \ldots T$:

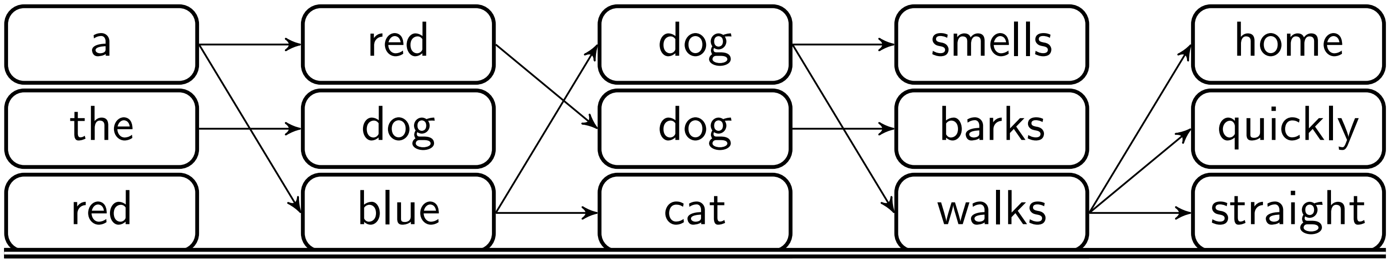
- For all k and for all possible output words w:

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)



For $t = 1 \dots T$:

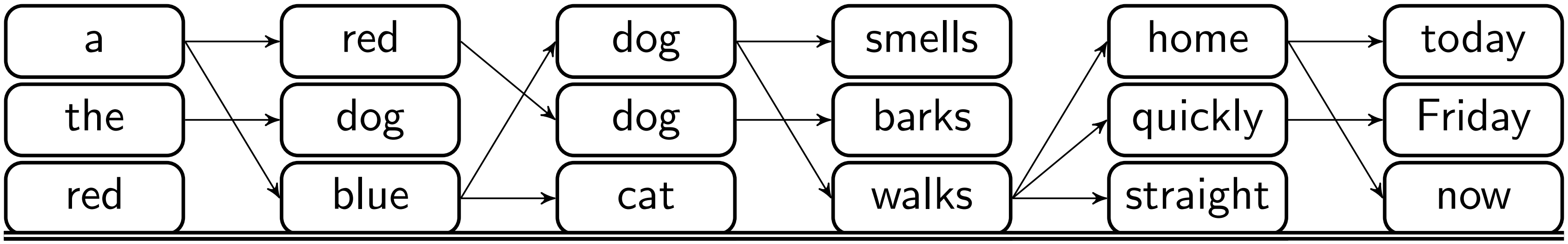
- For all k and for all possible output words w:

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

- Update beam:

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

Review: Beam Search at Test Time ($K = 3$)



For $t = 1 \dots T$:

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

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

- Update beam:

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

Seq2Seq Issues Revisited

Issue #1: Train/Test Mismatch (cf., **?**)

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

(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., **?**)

$$\mathsf{NLL}(\theta) = -\sum_t \ln p(w_t = y_t | y_{1:t-1}, \boldsymbol{x}; \theta)$$

(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., **?**)

$$\mathsf{NLL}(\theta) = -\sum_t \ln p(w_t = y_t | y_{1:t-1}, \boldsymbol{x}; \theta)$$

(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., **?**)

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

(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:

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

- $y_{1:t}$ 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:

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

- $y_{1:t}$ 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:

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

- $y_{1:t}$ 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 $\hat{y}_{1:t}^{(K)}$

Seq2Seq Issues Revisited

Issue #2: Seq2Seq models next-word probabilities:

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

(a) Sequence score is sum of locally normalized word-scores; gives rise to "Label Bias" [?]

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

Idea #2: Don't locally normalize

Seq2Seq Issues Revisited

Issue #2: Seq2Seq models next-word probabilities:

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

(a) Sequence score is sum of locally normalized word-scores; gives rise to "Label Bias" [?]

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

Idea #2: Don't locally normalize

Seq2Seq Issues Revisited

Issue #2: Seq2Seq models next-word probabilities:

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

(a) Sequence score is sum of locally normalized word-scores; gives rise to "Label Bias" [?]

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

Idea #2: Don't locally normalize

Seq2Seq Issues Revisited

Issue #2: Seq2Seq models next-word probabilities:

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

(a) Sequence score is sum of locally normalized word-scores; gives rise to "Label Bias" [**?**]

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

Idea #2: Don't locally normalize

Idea #2: Don't locally normalize

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

Idea #2: Don't locally normalize

$$s(w, \hat{y}_{1:t-1}^{(k)}) = \cancel{\ln p(\hat{y}_{1:t-1}^{(k)}|\boldsymbol{x}) + \ln \mathrm{softmax}(\mathbf{W}_{out}\,\mathbf{h}_{t-1}^{(k)} + \mathbf{b}_{out})}$$

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

Idea #2: Don't locally normalize

$$s(w, \hat{y}^{(k)}_{1:t-1}) = \cancel{\ln p(\hat{y}^{(k)}_{1:t-1} | \boldsymbol{x}) + \ln \mathrm{softmax}(\mathbf{W}_{out}\, \mathbf{h}^{(k)}_{t-1} + \mathbf{b}_{out})}$$

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

- Can set $s(w, \hat{y}^{(k)}_{1:t-1}) = -\infty$ if $(w, \hat{y}^{(k)}_{1:t-1})$ violates a hard constraint

Computing Gradients of the Loss ($K = 3$)

a

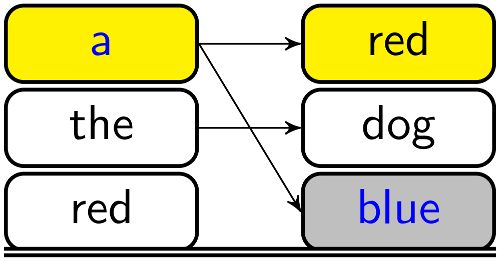
the

red

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

- Color Gold: target sequence y
- Color Gray: violating sequence $\hat{y}^{(K)}$

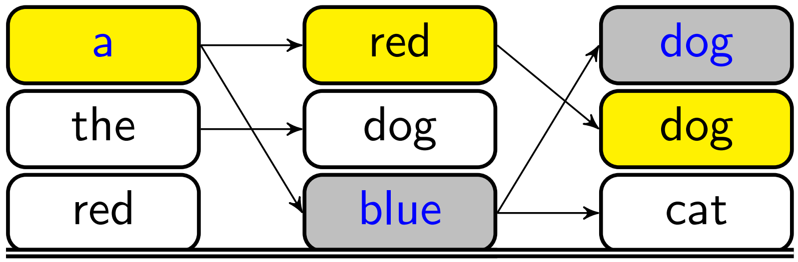
Computing Gradients of the Loss $(K = 3)$



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

- Color Gold: target sequence y
- Color Gray: violating sequence $\hat{y}^{(K)}$

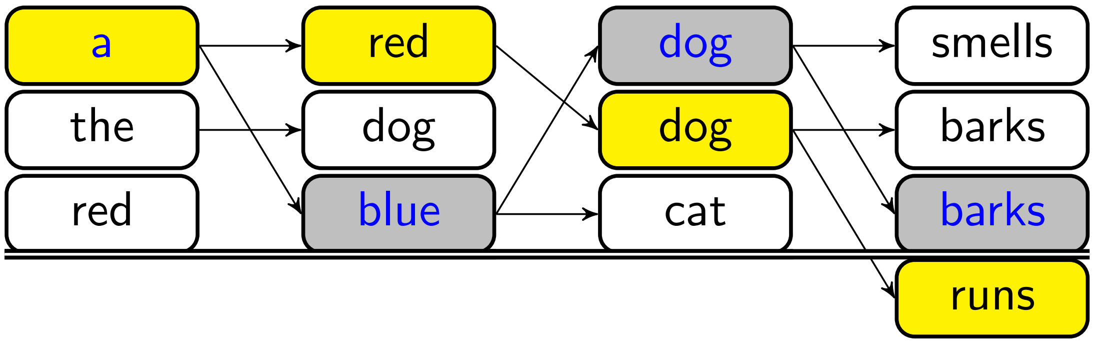
Computing Gradients of the Loss ($K = 3$)



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

- Color Gold: target sequence y
- Color Gray: violating sequence $\hat{y}^{(K)}$

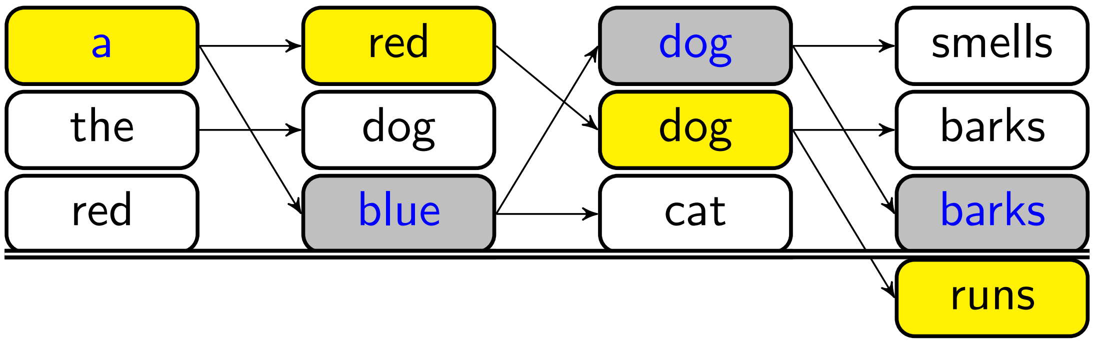
Computing Gradients of the Loss $(K = 3)$



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

- Color Gold: target sequence y
- Color Gray: violating sequence $\hat{y}^{(K)}$

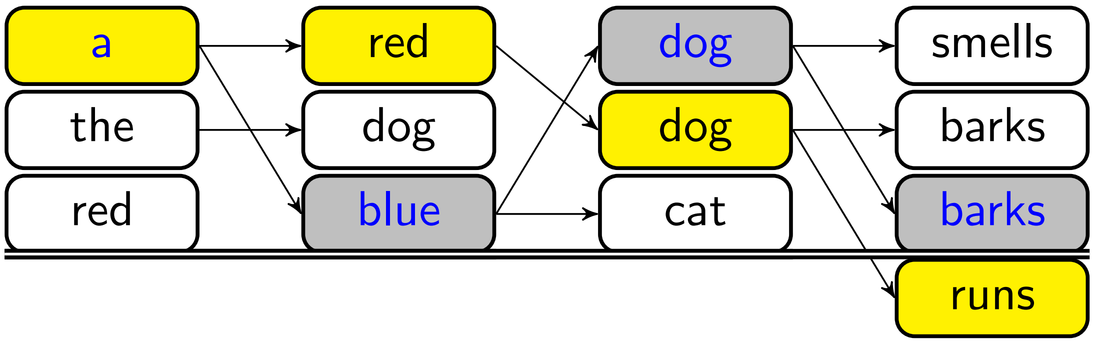
Computing Gradients of the Loss ($K = 3$)



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

- 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 [?] beam-update

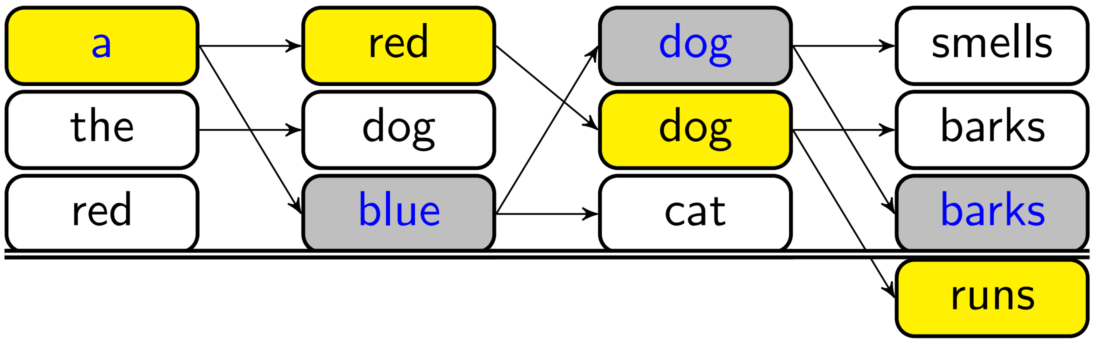
Computing Gradients of the Loss ($K = 3$)



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

- 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 [?] beam-update

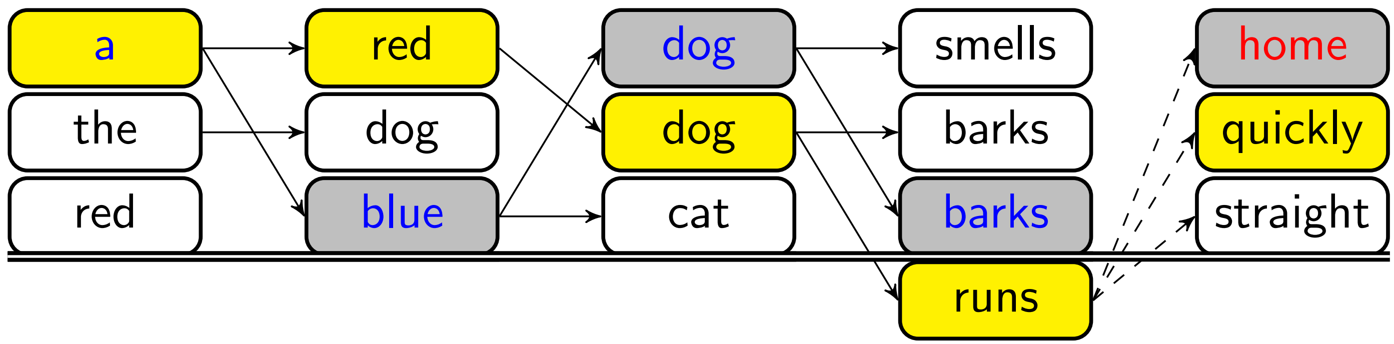
Computing Gradients of the Loss ($K = 3$)



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

- 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 [?] beam-update

Computing Gradients of the Loss $(K = 3)$

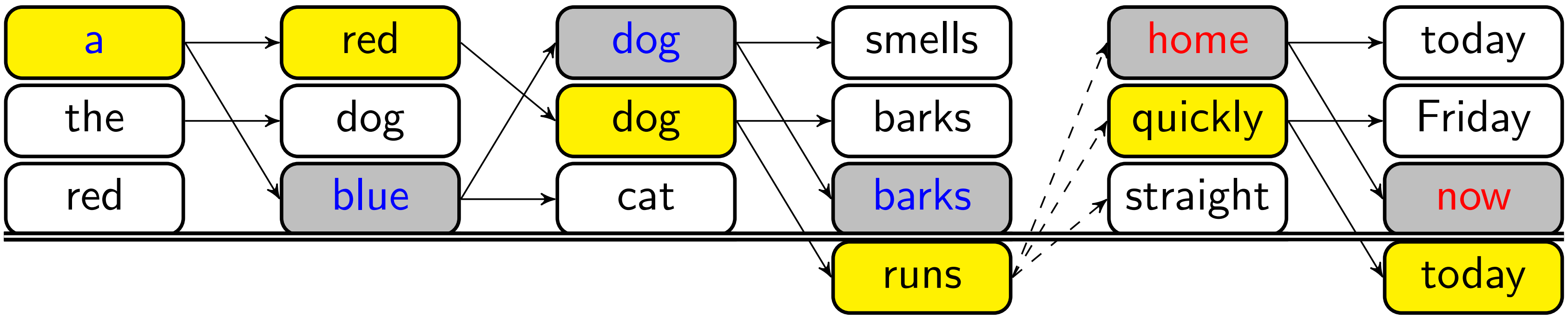


$$\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]$$

LaSO [?]:

- 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 ($K = 3$)

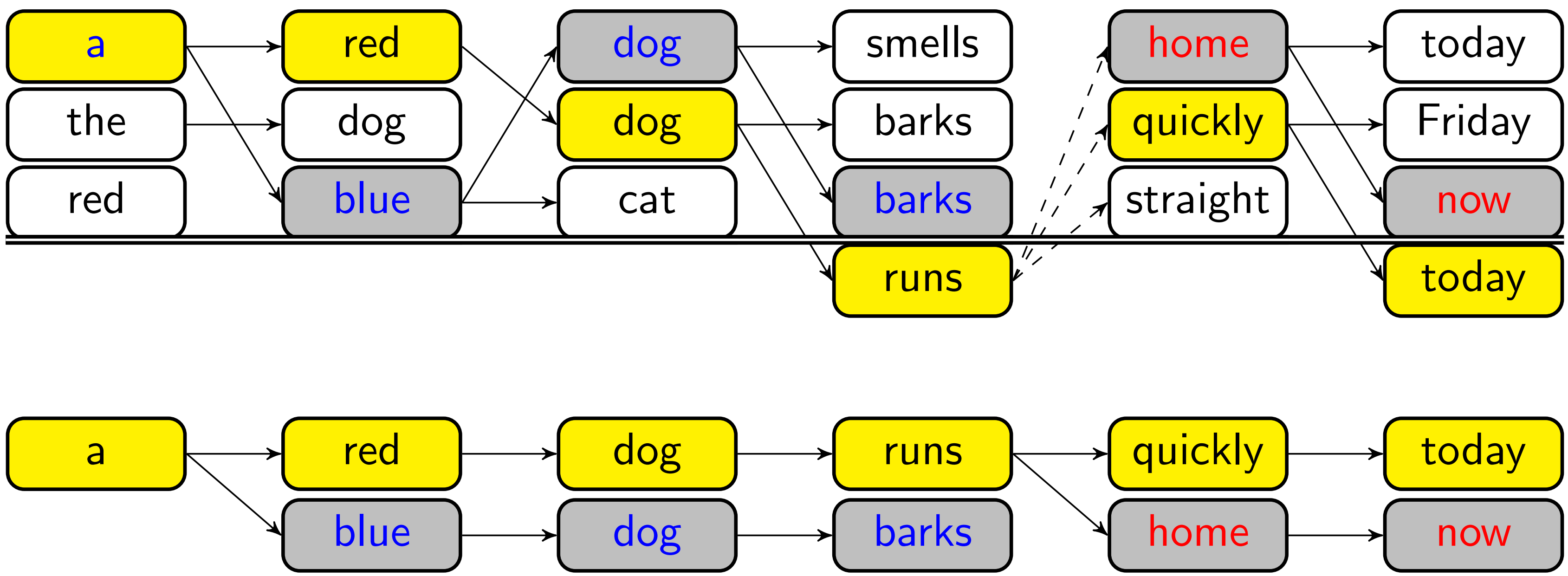


$$\mathcal{L}(\theta) = \sum_t \Delta(\hat{y}_{1:t}^{(K)}) \left[1 - s(y_t, y_{1:t-1}) + s(\hat{y}_t^{(K)}, \hat{y}_{1:t-1}^{(K)}) \right]$$

LaSO [?]:

- If no margin violation at $t-1$, update beam as usual
- Otherwise, update beam with sequences prefixed by $y_{1:t-1}$

Backpropagation over Structure



- 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 [**??**]
  - Globally Normalized Transition-Based Networks [**?**]
- RL-based approaches
  - MIXER [**?**]
  - 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 [**?**]
- Pre-trained with NLL; K increased gradually
- "BSO" uses unconstrained search; "ConBSO" uses constraints

[1] `https://github.com/harvardnlp/seq2seq-attn`

Word Ordering Experiments

	Word Ordering (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
Seq2Seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
ConBSO	**28.6**	**34.3**	**34.5**

- Map shuffled sentence to correctly ordered sentence
- Same setup as **?**
- BSO models trained with beam of size 6

Word Ordering Experiments

	Word Ordering (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
Seq2Seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
ConBSO	**28.6**	**34.3**	**34.5**

- Map shuffled sentence to correctly ordered sentence
- Same setup as **?**
- BSO models trained with beam of size 6

Word Ordering Experiments

	Word Ordering (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
Seq2Seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
ConBSO	**28.6**	**34.3**	**34.5**

- Map shuffled sentence to correctly ordered sentence
- Same setup as **?**
- 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

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

- BSO models trained with beam of size 6
- Same setup and evaluation as ?
- Certainly not SOA, but reasonable for word-only, left-to-right model

Machine Translation: Impact of Non-0/1 Δ

	Machine Translation (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
$\Delta(\hat{y}_{1:t}^{(k)}) = \mathbf{1}\{\text{margin violation}\}$	25.73	28.21	27.43
$\Delta(\hat{y}_{1:t}^{(k)}) = 1 - \mathrm{SentBLEU}(\hat{y}_{r+1:t}^{(K)}, y_{r+1:t})$	25.99	28.45	27.58

- 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

	Machine Translation (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
Seq2Seq	22.53	24.03	23.87
BSO	**23.83**	**26.36**	**25.48**
NLL	17.74	20.10	20.28
DAD [?]	20.12	22.25	22.40
MIXER/RL [?]	20.73	21.81	21.83

- IWSLT 2014, DE-EN
- BSO models trained with beam of size 6
- $\Delta(\hat{y}^{(k)}_{1:t}) = 1 - \mathrm{SentBLEU}(\hat{y}^{(K)}_{r+1:t}, y_{r+1:t})$
- Results in bottom sub-table from ?
- Note similar improvements to MIXER

Machine Translation Experiments

	Machine Translation (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
Seq2Seq	22.53	24.03	23.87
BSO	**23.83**	**26.36**	**25.48**
NLL	17.74	20.10	20.28
DAD [**?**]	20.12	22.25	22.40
MIXER/RL [**?**]	20.73	21.81	21.83

- IWSLT 2014, DE-EN
- BSO models trained with beam of size 6
- $\Delta(\hat{y}_{1:t}^{(k)}) = 1 - \mathrm{SentBLEU}(\hat{y}_{r+1:t}^{(K)}, y_{r+1:t})$
- Results in bottom sub-table from **?**
- 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

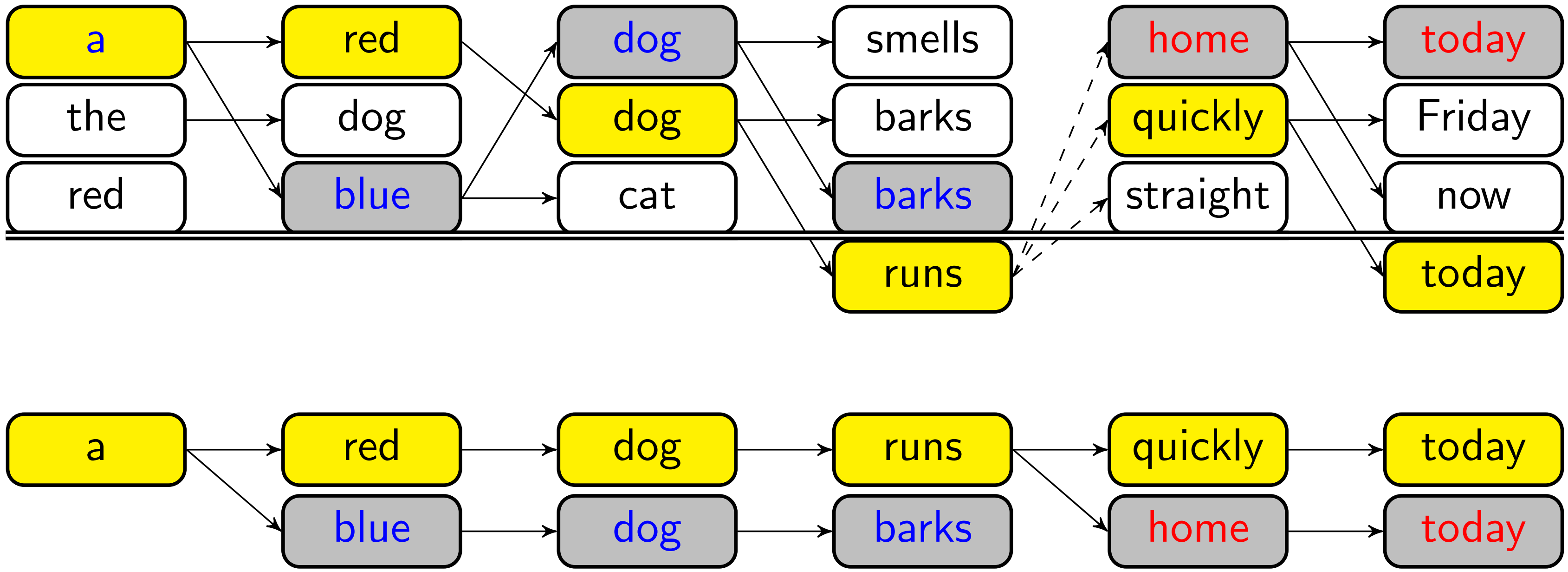
	Word Ordering Beam Size (BLEU)		
	$K_{te} = 1$	$K_{te} = 5$	$K_{te} = 10$
$K_{tr} = 2$	30.59	31.23	30.26
$K_{tr} = 6$	28.20	34.22	34.67
$K_{tr} = 11$	26.88	34.42	34.88

- ConBSO model, development set results

Pseudocode

procedure $\mathrm{BSO}(\boldsymbol{x}, K_{tr}, \mathrm{succ})$
  Init empty storage $\hat{y}_{1:T}$ and $\hat{\boldsymbol{h}}_{1:T}$; init S_1
  $r \leftarrow 0$; $violations \leftarrow \{0\}$
  for $t = 1, \ldots, T$ **do** ▷ Forward
    $K = K_{tr}$ if $t \neq T$ else $\underset{k: \hat{y}^{(k)}_{1:t} \neq y_{1:t}}{\arg\max} f(\hat{y}^{(k)}_t, \hat{\boldsymbol{h}}^{(k)}_{t-1})$
    if $f(y_t, \boldsymbol{h}_{t-1}) < f(\hat{y}^{(K)}_t, \hat{\boldsymbol{h}}^{(K)}_{t-1}) + 1$ **then**
      $\hat{\boldsymbol{h}}_{r:t-1} \leftarrow \hat{\boldsymbol{h}}^{(K)}_{r:t-1}$
      $\hat{y}_{r+1:t} \leftarrow \hat{y}^{(K)}_{r+1:t}$
      Add t to $violations$; $r \leftarrow t$
      $S_{t+1} \leftarrow \mathrm{topK}(\mathrm{succ}(y_{1:t}))$
    else
      $S_{t+1} \leftarrow \mathrm{topK}(\bigcup_{k=1}^{K} \mathrm{succ}(\hat{y}^{(k)}_{1:t}))$
  $grad\_\boldsymbol{h}_T \leftarrow \mathbf{0}$; $grad\_\widehat{\boldsymbol{h}}_T \leftarrow \mathbf{0}$
  for $t = T-1, \ldots, 1$ **do** ▷ Backward
    $grad\_\boldsymbol{h}_t \leftarrow \mathrm{BRNN}(\nabla_{\boldsymbol{h}_t} \mathcal{L}_{t+1}, grad\_\boldsymbol{h}_{t+1})$
    $grad\_\widehat{\boldsymbol{h}}_t \leftarrow \mathrm{BRNN}(\nabla_{\widehat{\boldsymbol{h}}_t} \mathcal{L}_{t+1}, grad\_\widehat{\boldsymbol{h}}_{t+1})$
    if $t-1 \in violations$ **then**
      $grad\_\boldsymbol{h}_t \leftarrow grad\_\boldsymbol{h}_t + grad\_\widehat{\boldsymbol{h}}_t$
      $grad\_\widehat{\boldsymbol{h}}_t \leftarrow \mathbf{0}$

Backpropagation over Structure



- Margin gradients are sparse, only violating sequences get updates.
- Backprop only requires 2x time as standard methods.