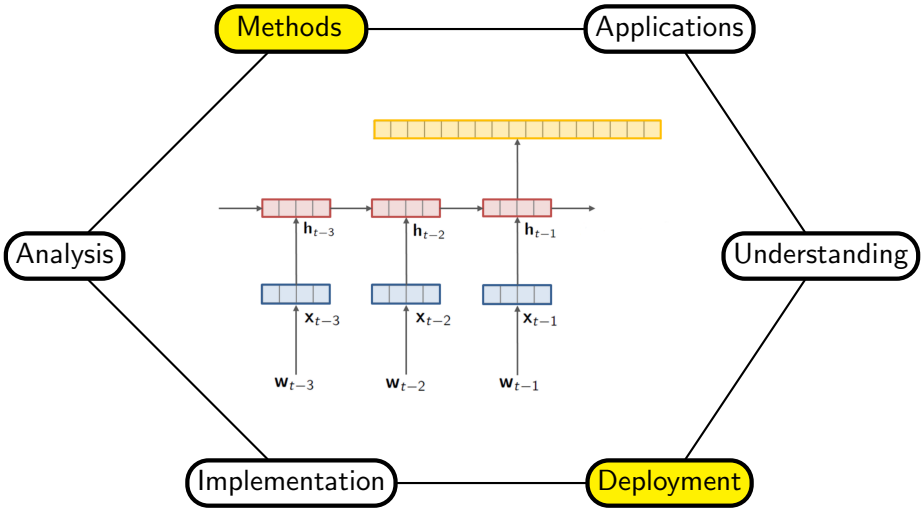
Learning How to Say It:
Language Generation post Deep Learning

Alexander M Rush

Research Direction



Structured Modeling

Generation Setup (Reminder)

$$y^*_{1:T} = \arg\max_{y_{1:T}} f(y_{1:T}; x, \theta)$$

- Input $x_{1:S}$, *what to talk about*
- Output text $y^*_{1:T}$, *how to say it*
- Model $f(.;\theta)$, learned from data

Generation Setup (Reminder)

$$y^*_{1:T} = \arg\max_{y_{1:T}} f(y_{1:T}; x, \theta)$$

- Input $x_{1:S}$, *what to talk about*
- Output text $y^*_{1:T}$, *how to say it*
- Model $f(.;\theta)$, learned from data

Generation Setup (Reminder)

$$y^*_{1:T} = \arg\max_{y_{1:T}} f(y_{1:T}; x, \theta)$$

- Input $x_{1:S}$, *what to talk about*
- Output text $y^*_{1:T}$, *how to say it*
- Model $f(.;\theta)$, learned from data

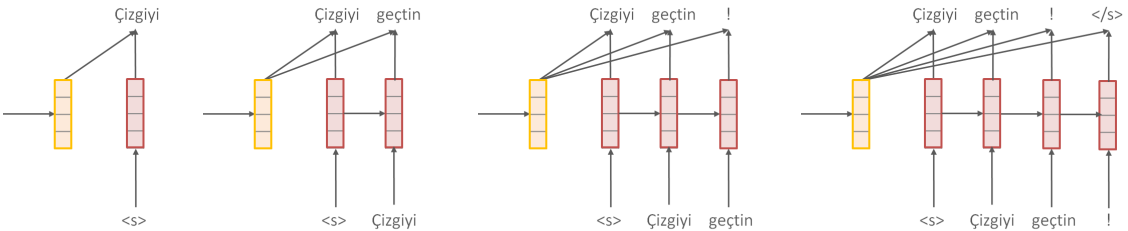
Generation Setup (Reminder)

$$y^*_{1:T} = \arg\max_{y_{1:T}} f(y_{1:T}; x, \theta)$$

- Input $x_{1:S}$, *what to talk about*
- Output text $y^*_{1:T}$, *how to say it*
- Model $f(.;\theta)$, learned from data

Training Seq2Seq

Parameters θ are trained to predict the next word *given the true history.*

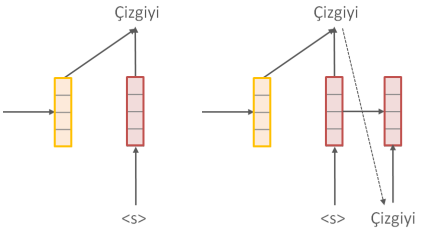


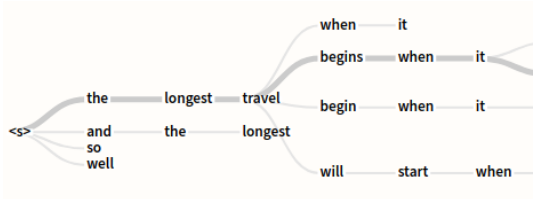
Pretend it is Multiclass classification:

$$\mathsf{NLL}(\theta) = -\sum_{t} \log p(y_t | y_{1:t-1}, \mathbf{c}; \theta)$$

Deploying Seq2Seq

Parameters θ is deployed to predict a next word *given the predicted history*.

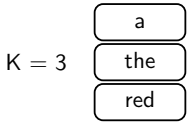




Deploy as a structured model:

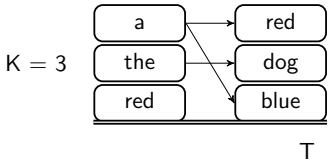
$$y^*_{1:T} = \arg\max_{y_{1:T}} f(y_{1:T}; \theta) = \arg\max_{y_{1:T}} \sum_t \log p(y_t | y_{1:t-1}, \mathbf{c}; \theta)$$

Best Sequence Heuristic: Beam Search

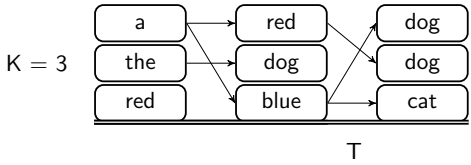


T

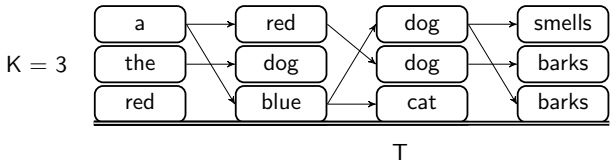
Best Sequence Heuristic: Beam Search



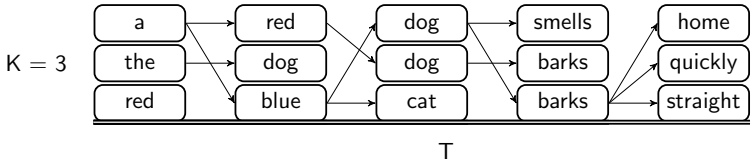
Best Sequence Heuristic: Beam Search



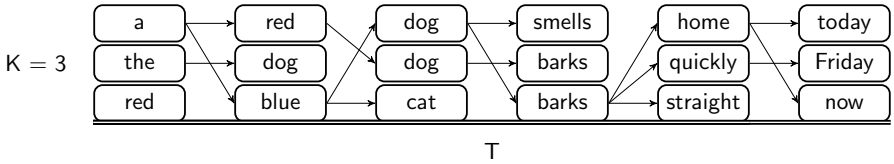
Best Sequence Heuristic: Beam Search



Best Sequence Heuristic: Beam Search



Best Sequence Heuristic: Beam Search



1. Compute the score of every possible next word.

$$f(y_t, y_{1:t-1}^{(k)}) \leftarrow \log p(y_t \mid y_{1:t-1}^{(k)}, \mathbf{c}) + \log p(y_{1:t-1}^{(k)} \mid \mathbf{c})$$

2. Prune to only the K highest-scoring,

$$y_{1:t}^{(1:K)} \leftarrow K \arg\max_{y_{1:t}} f(y_t, y_{1:t-1}^{(k)})$$

Theoretical Issues

1. Exposure Bias
   - Training by conditioning on true $y_{1:t-1}$.
2. Metric Bias
   - Training with local NLL, evaluate with hamming-style losses.
3. Label Bias
   - Locally normalized models have known pathological issues.

Work

Can we exploit discrete sequences to improve models for text generation?

Applications:

- (1) Sequence-to-Sequence as Beam Search Optimization for training.
- (2) Sequence Knowledge Distillation for deployment.

Beam Search Optimization (Wiseman and Rush [2016])

Motivation: Can we fix target theoretical issues by unifying training and test objective?

Change 1: Modify Scoring Function

Same model, but replace $\log p(y_t | y_{1:t-1}^{(k)}, \mathbf{c}; \theta)$ with globally normalized $f(y_t, y_{1:t-1}^{(k)}, \mathbf{c}; \theta)$

Change 2: Run Beam Search During Training

1. Compute the score of every possible next word.

$$f(y_t, y_{1:t-1}^{(k)}) \leftarrow \log p(y_t \mid y_{1:t-1}^{(k)}, \mathbf{c}) + \log p(y_{1:t-1}^{(k)} \mid \mathbf{c})$$

2. Prune to only the K highest-scoring,

$$y_{1:t}^{(1:K)} \leftarrow K \arg\max_{y_{1:t}} f(y_t, y_{1:t-1}^{(k)})$$

Change 2: Run Beam Search During Training

1. Compute the score of every possible next word.

$$f(y_t, y_{1:t-1}^{(k)}) \leftarrow f(y_t, y_{1:t-1}^{(k)}, \mathbf{c}; \theta)$$

2. Prune to only the K highest-scoring,

$$y_{1:t}^{(1:K)} \leftarrow K \arg\max_{y_{1:t}} f(y_t, y_{1:t-1}^{(k)})$$

Change 3: Replace train to enforce beam-search margin

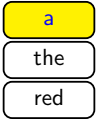
New Global Objective:

- Margin between gold seq $y^{(g)}$ and last seq on beam $y^{(K)}$

$$\mathcal{L}(\theta) = \sum_t \Delta(y^{(g)}_{1:t}, y^K_{1:t}) \left[1 - f(y^{(g)}_t, y^{(g)}_{1:t-1}, \mathbf{c}) + f(y^{(K)}_t, y^{(K)}_{1:t-1}, \mathbf{c})\right]$$

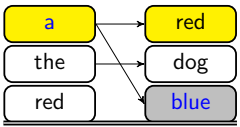
- Slack-rescaled, margin-based sequence criterion, at each time step.
- When violation occurs, target replaces current beam (learning as search optimization **?**)

Beam Search Optimization Training Example



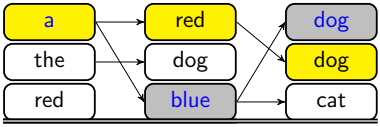
- Color Gold: target sequence $y^{(g)}$
- Color Gray: violating sequence $y^{(K)}$

Beam Search Optimization Training Example



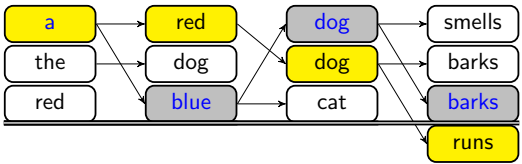
- Color Gold: target sequence $y^{(g)}$
- Color Gray: violating sequence $y^{(K)}$

Beam Search Optimization Training Example



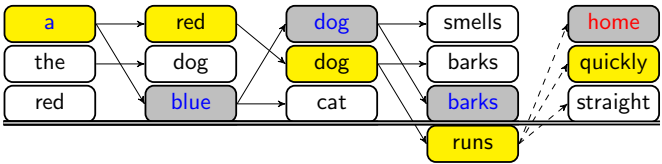
- Color Gold: target sequence $y^{(g)}$
- Color Gray: violating sequence $y^{(K)}$

Beam Search Optimization Training Example



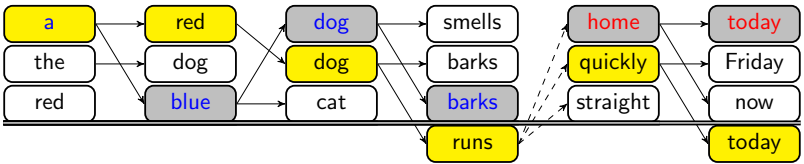
- Color Gold: target sequence $y^{(g)}$
- Color Gray: violating sequence $y^{(K)}$

Beam Search Optimization Training Example



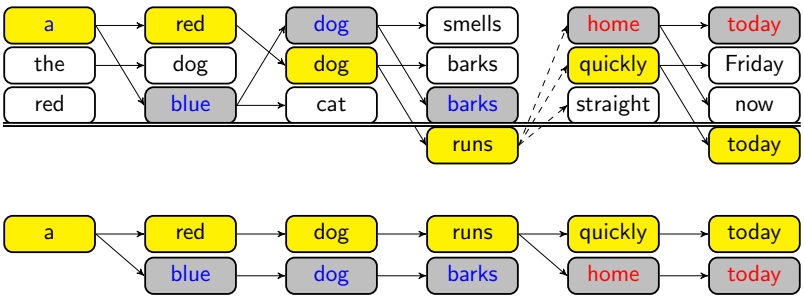
- Color Gold: target sequence $y^{(g)}$
- Color Gray: violating sequence $y^{(K)}$

Beam Search Optimization Training Example



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

Structured Backpropagation



- Margin gradients are sparse, only violating sequences get updates.
- Backprop as efficient as standard models.

Theoretical Issues Revisited

- Exposure Bias
  - Beam search at training
- Train/Test Loss Mismatch
  - Slack-rescaled margin can capture correct loss.
- Label Bias **?**
  - Sequence regression is not locally normalized

	$K_e = 1$	$K_e = 5$	$K_e = 10$
	Word Ordering (BLEU)		
seq2seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
BSO-Con	**28.6**	**34.3**	**34.5**

	$K_e = 1$	$K_e = 5$	$K_e = 10$
	Word Ordering (BLEU)		
seq2seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
BSO-Con	**28.6**	**34.3**	**34.5**
	Dependency Parsing (UAS/LAS)		
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**
BSO-Con	85.11/79.32	**91.25**/86.92	**91.57**/87.26

	$K_e = 1$	$K_e = 5$	$K_e = 10$
	Word Ordering (BLEU)		
seq2seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
BSO-Con	**28.6**	**34.3**	**34.5**
	Dependency Parsing (UAS/LAS)		
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**
BSO-Con	85.11/79.32	**91.25**/86.92	**91.57**/87.26
	Machine Translation (BLEU)		
seq2seq	22.53	24.03	23.87
BSO, SB-Δ, K_t=6	**23.83**	**26.36**	**25.48**
XENT	17.74	$\leq$ 20.5	$\leq$ 20.5
DAD	20.12	$\leq$ 22.5	$\leq$ 23.0
MIXER	20.73	-	$\leq$ 22.0

Sequence Knowledge Distillation (Kim and Rush [2016])

Goal: Compress text generation models.

- **Pruning**: Prune weights based on importance criterion **??**
- **Knowledge Distillation**: Train a *student* model to learn from a *teacher* model **???**.

Other methods:

- low-rank matrix factorization of weight matrices **?**
- weight binarization **?**
- weight sharing **?**

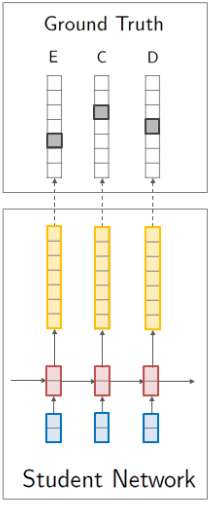
Baseline Model

Standard model minimize NLL(θ):

$$-\sum_{t} \log p(y_t = y_t^{(g)} \mid y_{1:t-1}^{(g)}, \mathbf{c}; \theta)$$

where $y_t^{(g)}$ is the ground truth word at time t.

Cross-entropy with ground truth.

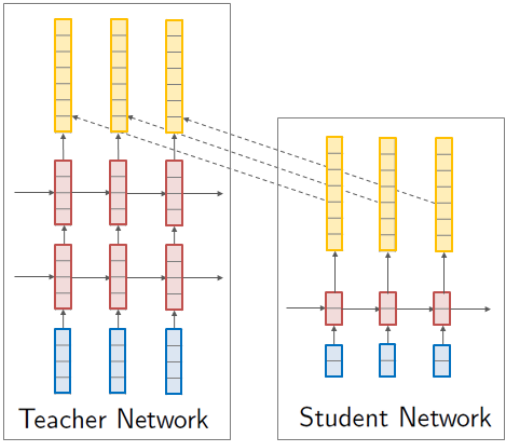


Standard Method: Word-Level Knowledge Distillation

Teacher network: $q(y_t|y_{1:t-1}, \mathbf{c}; \theta_T)$

Minimize cross-entropy between teacher and student distribution $\mathcal{L}_{\text{WORD-KD}}(\theta)$

$$-\sum_t \sum_v q(y_t = v \mid y^{(g)}_{1:t-1}, \mathbf{c}; \theta_T) \times \log p(y_t = v \mid y^{(g)}_{1:t-1}, \mathbf{c}; \theta)$$



Sequence-Level Knowledge Distillation

Proposal: Replace multi-class with sequence distribution. Instead of word NLL,

$$-\sum_t \sum_v q(y_t = v \mid y^{(g)}_{1:t-1}, \mathbf{c}; \theta_T) \times \log p(y_t = v \mid y^{(g)}_{1:t-1}, \mathbf{c}; \theta)$$

Minimize cross-entropy between q and p implied sequence-distribution

$$-\sum_{v_1} \dots \sum_{v_T} q(y_{1:T} = v_{1:T} | \mathbf{c}; \theta_T) \times \log p(y_{1:T} = v_{1:T} | \mathbf{c}; \theta)$$

However, as before this term is intractable.

A Simple Approximation

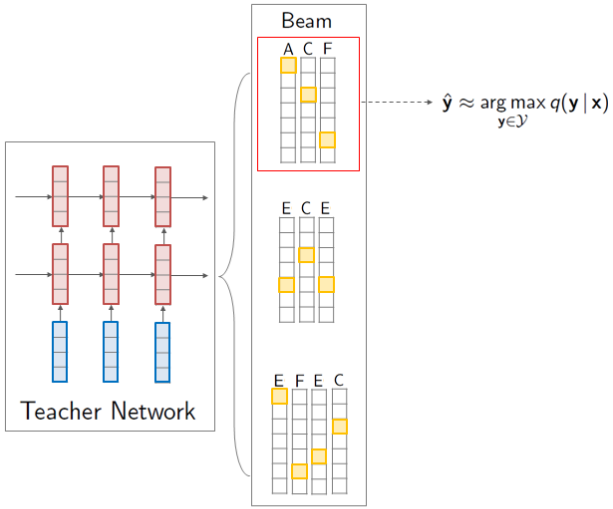
Approximate $q(y_{1:T} \mid \mathbf{c})$ with mode

$$q(y_{1:T} \mid \mathbf{c}) \approx \mathbf{1}\{\arg\max_{y} q(y_{1:T} \mid \mathbf{c})\}$$

Roughly obtained with beam search

$$y^*_{1:T} \approx \arg\max_{y_{1:T}} q(y_{1:T} \mid \mathbf{c})$$

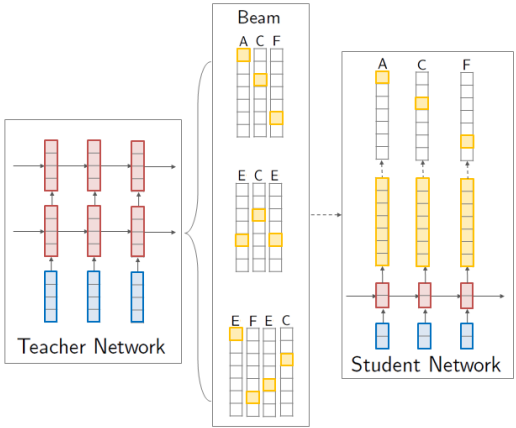
Empirically, point estimate captures significant mass



Sequence-Level Knowledge Distillation

$$\mathcal{L}_{\text{SEQ-KD}}(\theta) = -\log p(y^*_{1:T} \mid \mathbf{c}; \theta)$$
$$\approx -\sum_{v_{1:T}} q(y_{1:T} = v_{1:t} | \mathbf{c}; \theta_T) \log p(y_{1:T} | \mathbf{c}; \theta)$$

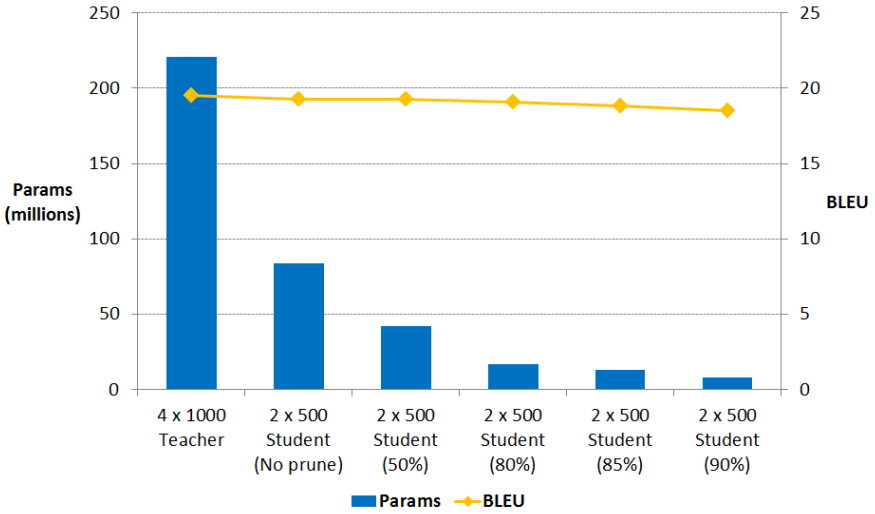
Simplest model: train the student model on y^* with NLL



Results: English → German

Model	BLEU$_{K=1}$	$\Delta_{K=1}$	BLEU$_{K=5}$	$\Delta_{K=5}$	PPL	$p(y^*)$
4×1000						
Teacher	17.7	–	19.5	–	6.7	1.3%
Seq-Inter	19.6	+1.9	19.8	+0.3	10.4	8.2%
2×500						
Student	14.7	–	17.6	–	8.2	0.9%
Word-KD	15.4	+0.7	17.7	+0.1	8.0	1.0%
Seq-KD	18.9	**+4.2**	19.0	+1.4	22.7	16.9%
Seq-Inter	18.9	**+4.2**	19.3	**+1.7**	15.8	7.6%

Combining Knowledge Distillation and Pruning



Application

Temporary

Yuntian Deng, Anssi Kanervisto, and Alexander M. Rush. 2016. What You Get Is What You See: A Visual Markup Decompiler. In *Arxiv*.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9735–9747.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. abs/1702.00887.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. In *AAAI*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *EMNLP*.

Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. 2018. Semi-amortized variational autoencoders.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017.

Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, System Demonstrations*, pages 67–72.

Brandon Reagen, Udit Gupta, Robert Adolf, Michael M Mitzenmacher, Alexander M Rush, Gu-Yeon Wei, and David Brooks. 2017. Weightless: Lossy weight encoding for deep neural network compression. *arXiv preprint arXiv:1711.04686*.

Alexander Rush. 2018. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*, September, pages 379–389.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart M. Shieber. 2016. Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction. In *arxiv*.

Jean Senellart, Dakun Zhang, WANG Bo, Guillaume Klein, Jean-Pierre Ramatchandirin,

Josep Crego, and Alexander Rush. 2018. Opennmt system description for wnmt 2018: 800 words/sec on a single-core cpu. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 122–128.

Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. 2019. Seq2seq-v is: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics*, 25(1):353–363.

Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. 2016. Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. In *Arxiv*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *EMNLP*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2017a. Challenges in Data-to-Document Generation. In *EMNLP*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017b. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*.