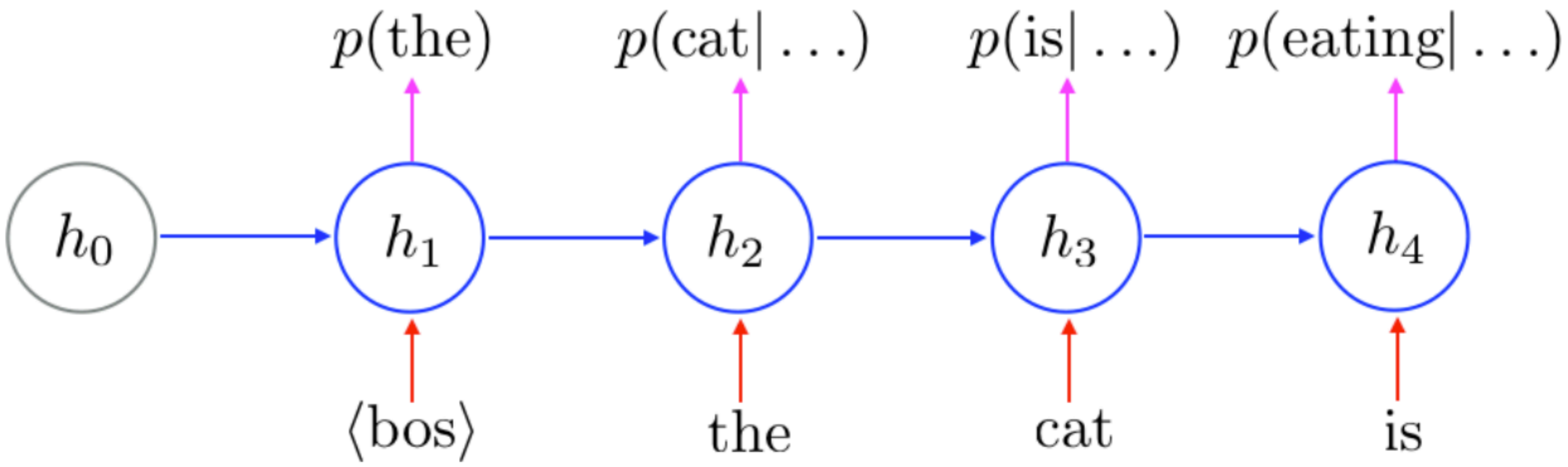
Survey on Optimizing Beam Search for Seq2Seq Learning

Yilin Yang

Table of Contents

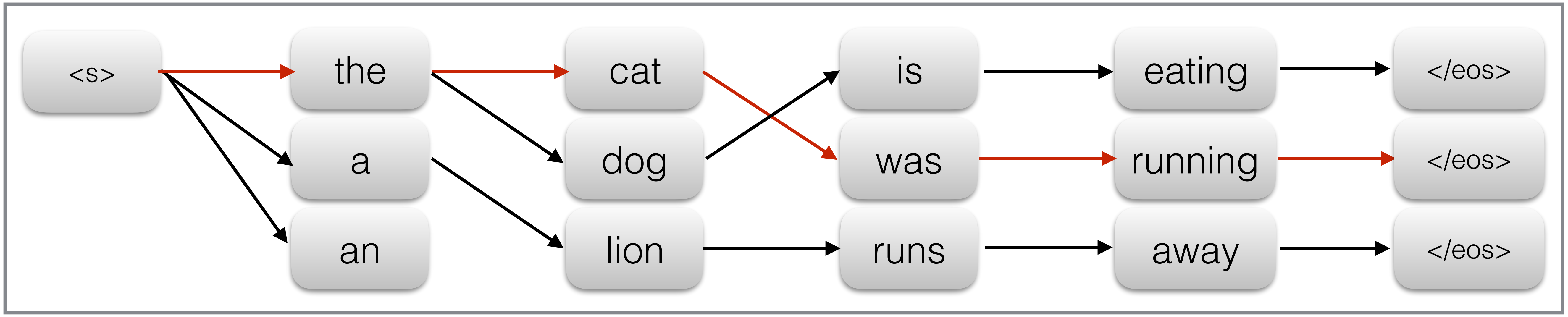
NMT Decoding

Example: $p(\text{the}, \text{cat}, \text{is}, \text{eating})$



NMT model could give you the probability of any target sequences.

Beam Search on NMT



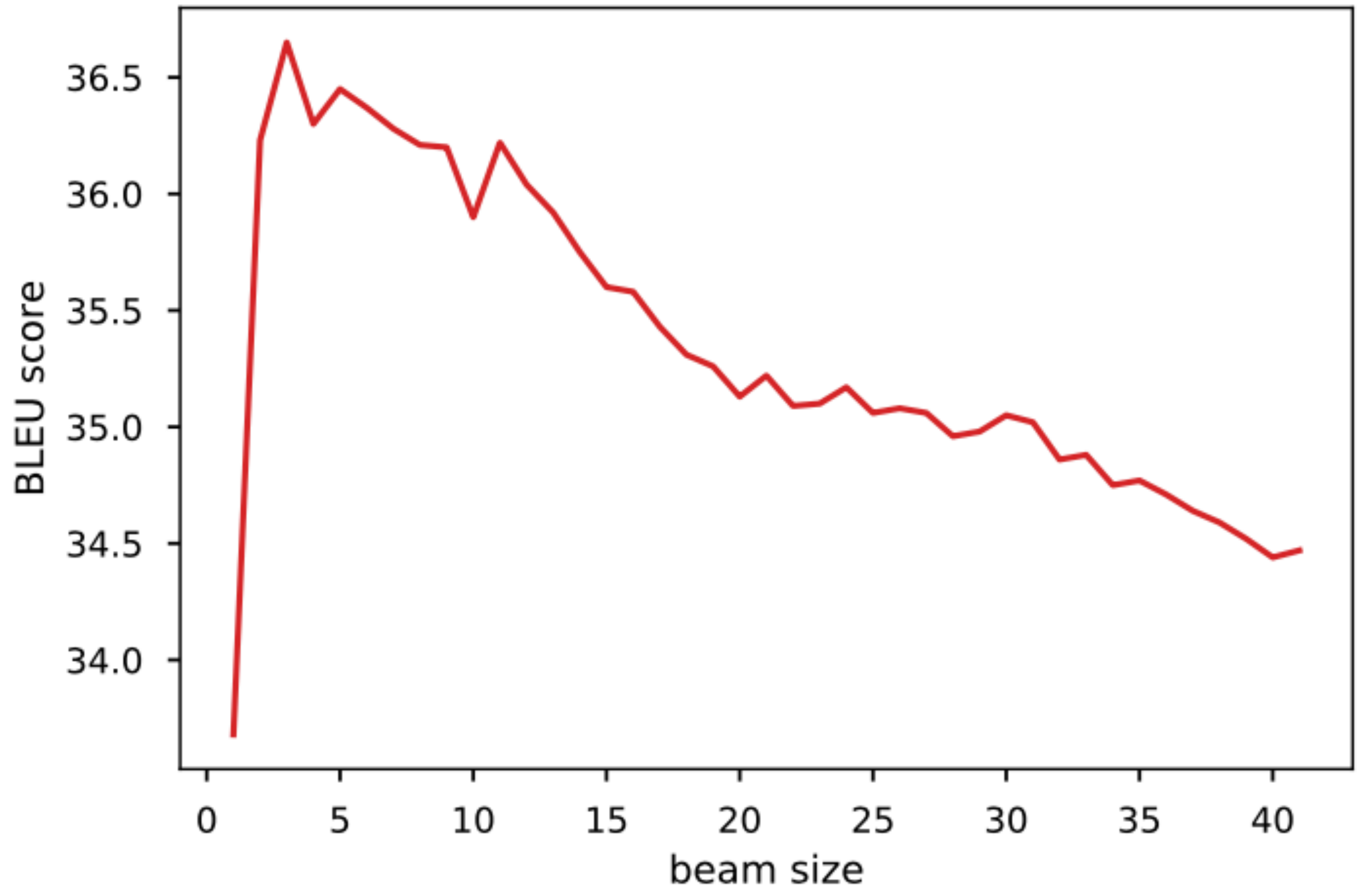
Step1	Step2	Step3	Step4	Log-Prob
the	dog	is	eating	-5
the	cat	was	running	-8
a	lion	runs	away	-15

Beam search covers larger search space.

Beam Size Heuristics

- When b = 1, beam-search becomes greedy search
- When b = $+\infty$, beam-search becomes global search (breadth-first search)

Beam Search Curse

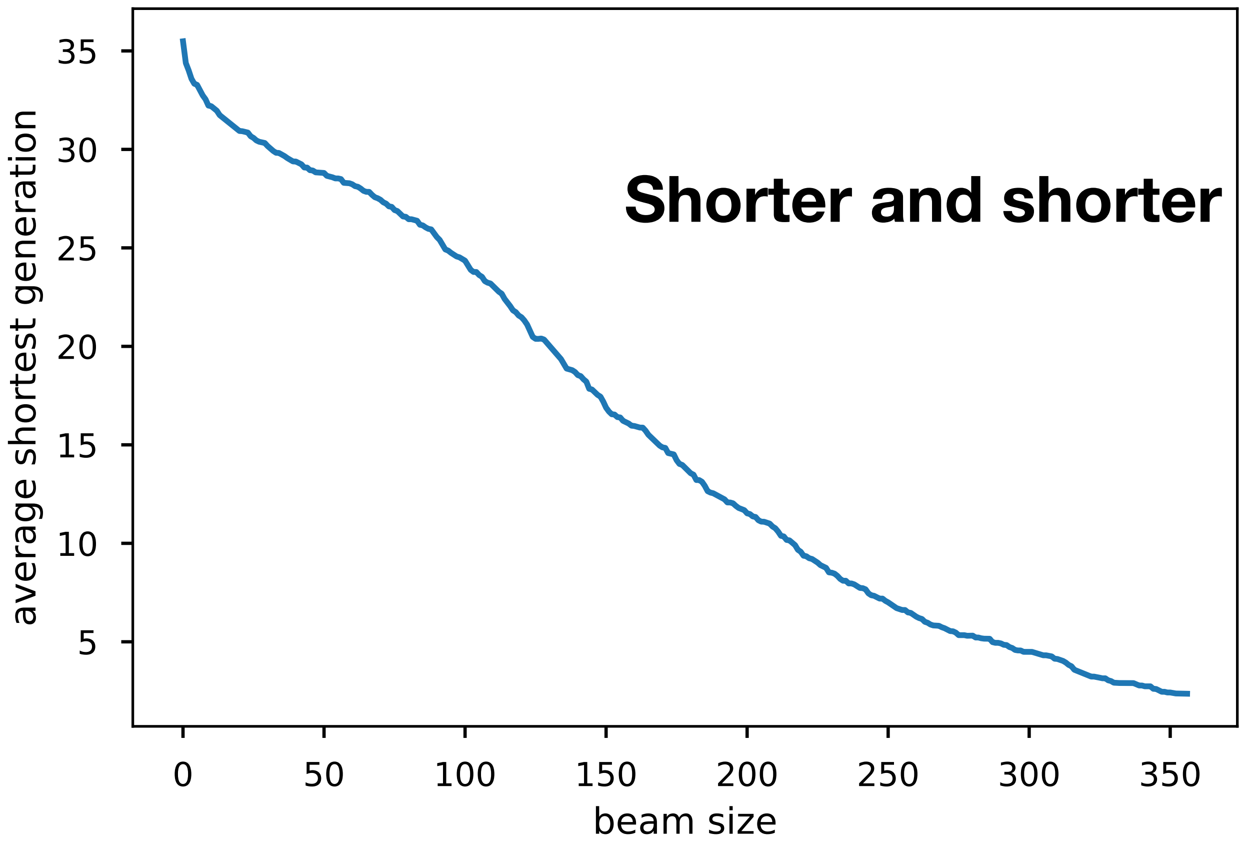


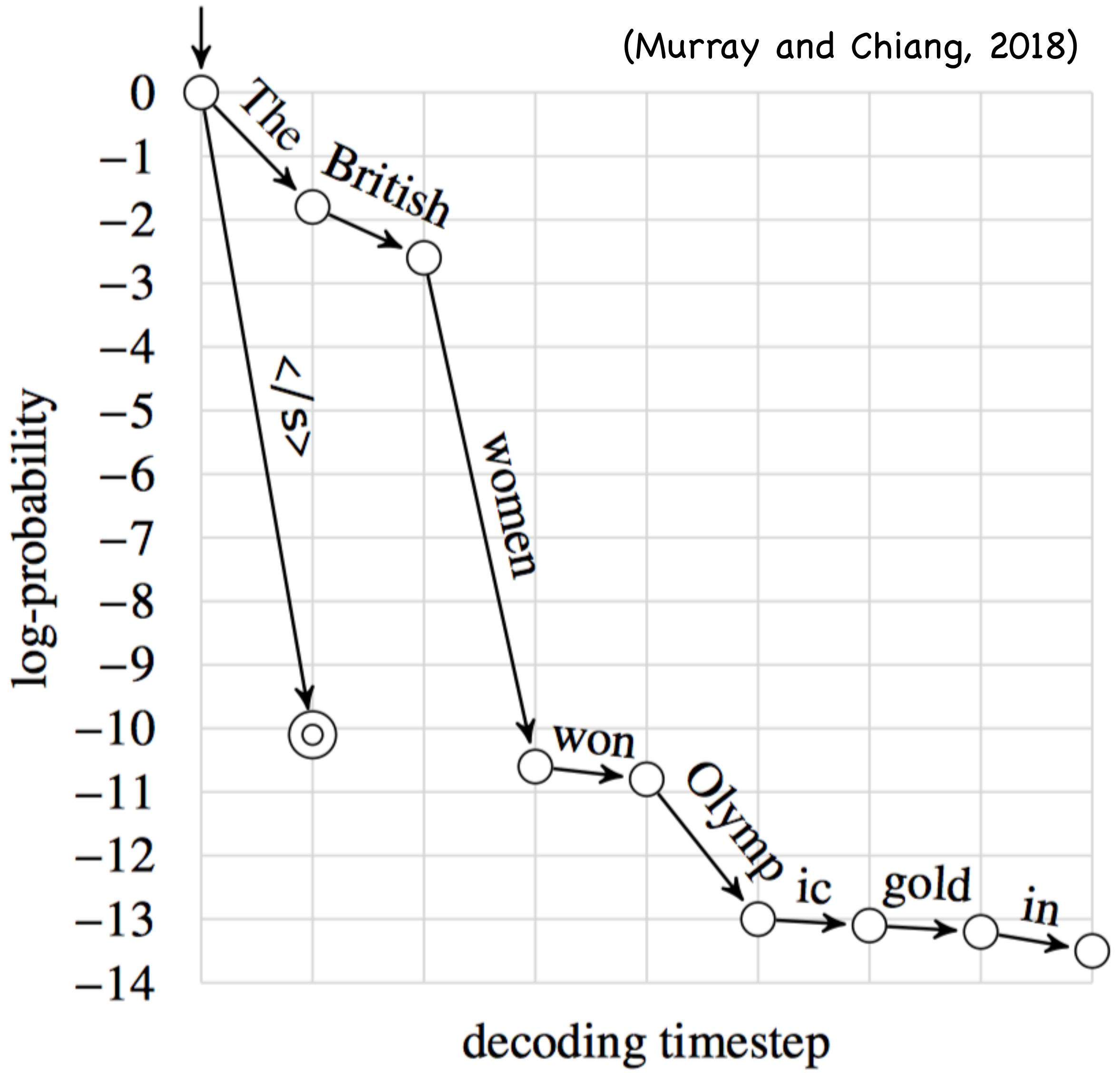
Seq2seq model couldn’t benefit from larger beam sizes

Explanation

- The beam search curse is due to following reasons
  - As beam size increases, the candidate lengths become shorter
  - Log-prob prefers shorter candidates
  - Thus, the generation becomes shorter, and faces harsh brevity penalty from BLEU.

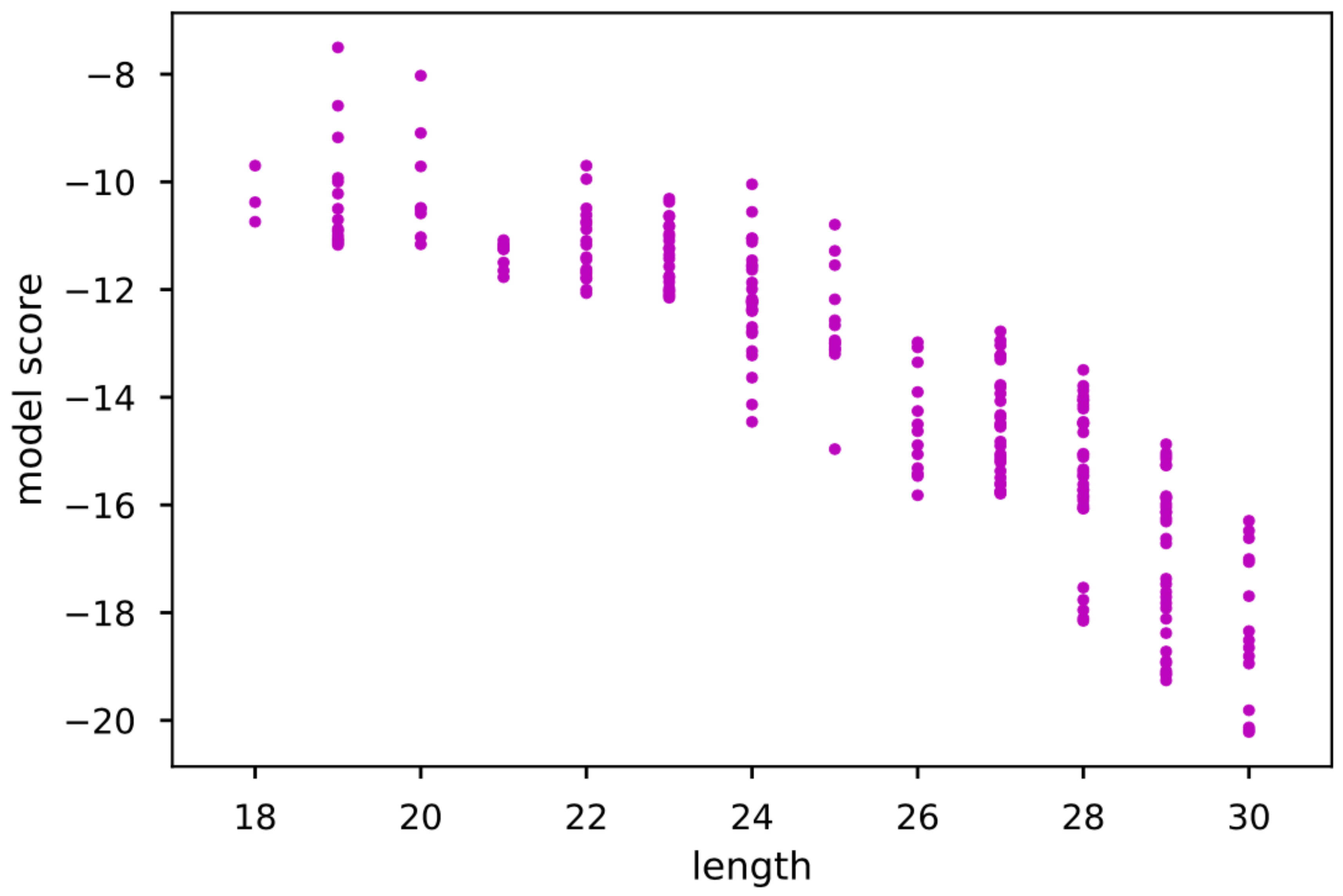
Explanation





With increasing beam sizes, beam search would generate shorter candidates

Explanation



Log-prob prefers shorter candidates

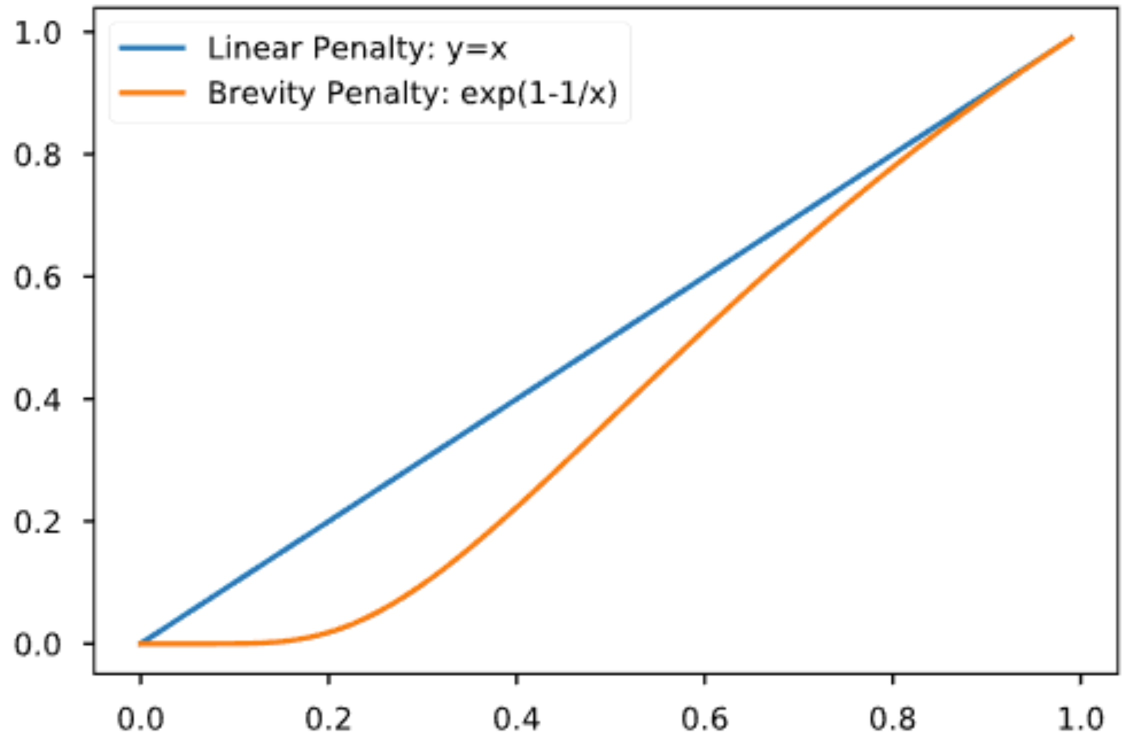
Explanation

$$\text{BLEU} = bp \cdot \exp(1/4 \textstyle\sum_{n=1}^{4} \log p_n) \qquad (2)$$

$$\text{where } \; bp = \min\{e^{1-1/lr}, 1\} \qquad (3)$$

$$\text{where } \; lr = |\mathbf{y}|/|\mathbf{y}^*| \qquad (4)$$

Here p_n are the n-gram precisions, and $|\mathbf{y}|$ and $|\mathbf{y}^*|$ denote the hypothesis and reference lengths,



- As generation length decreases, BLEU would penalize it slightly harsher than linearly.

Approaches

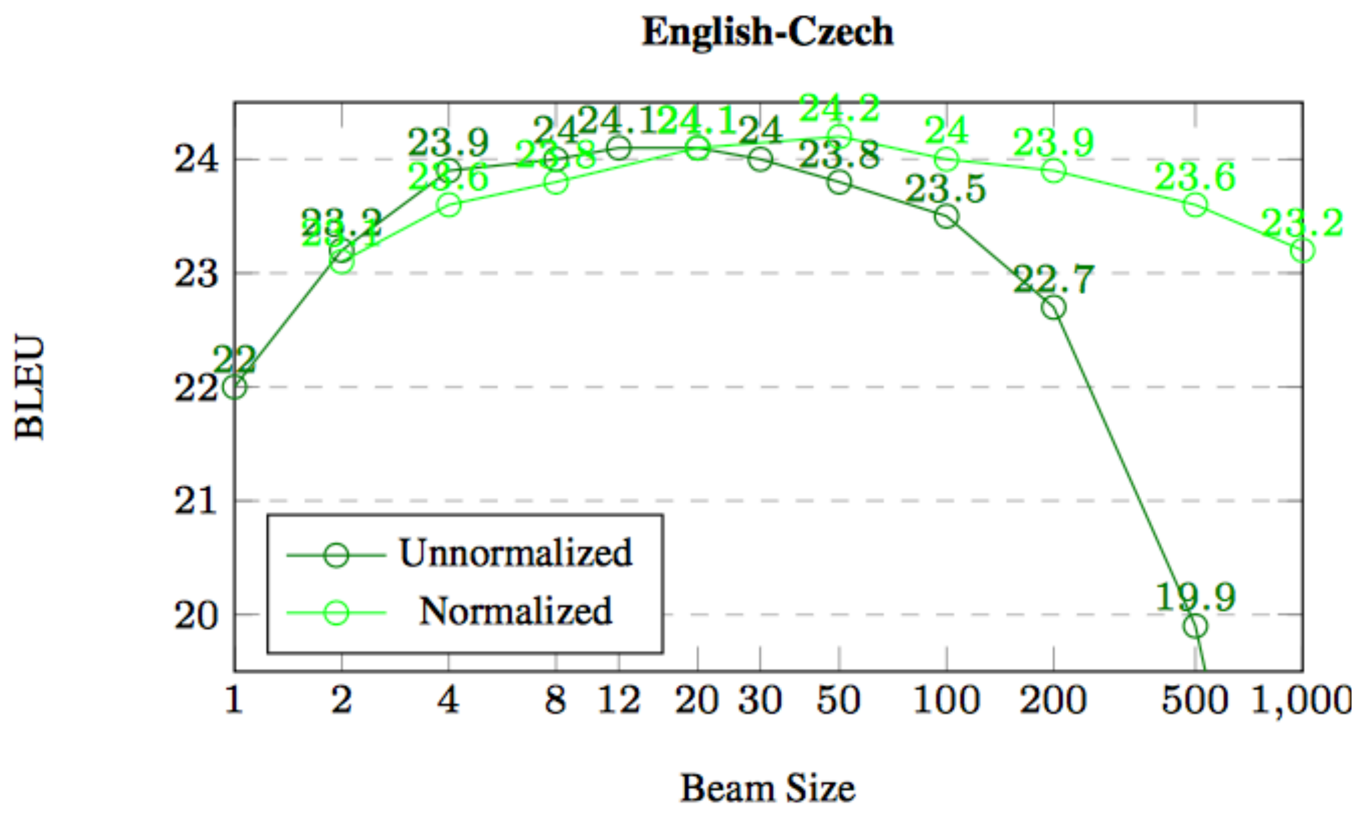
- Break the beam search curse by a better reranking method than log-prob, to take generation length into account.
  - Length normalization
  - Our approaches:
    - Bounded word-reward
    - Bounded adaptive-reward
    - Brevity-penalty normalization

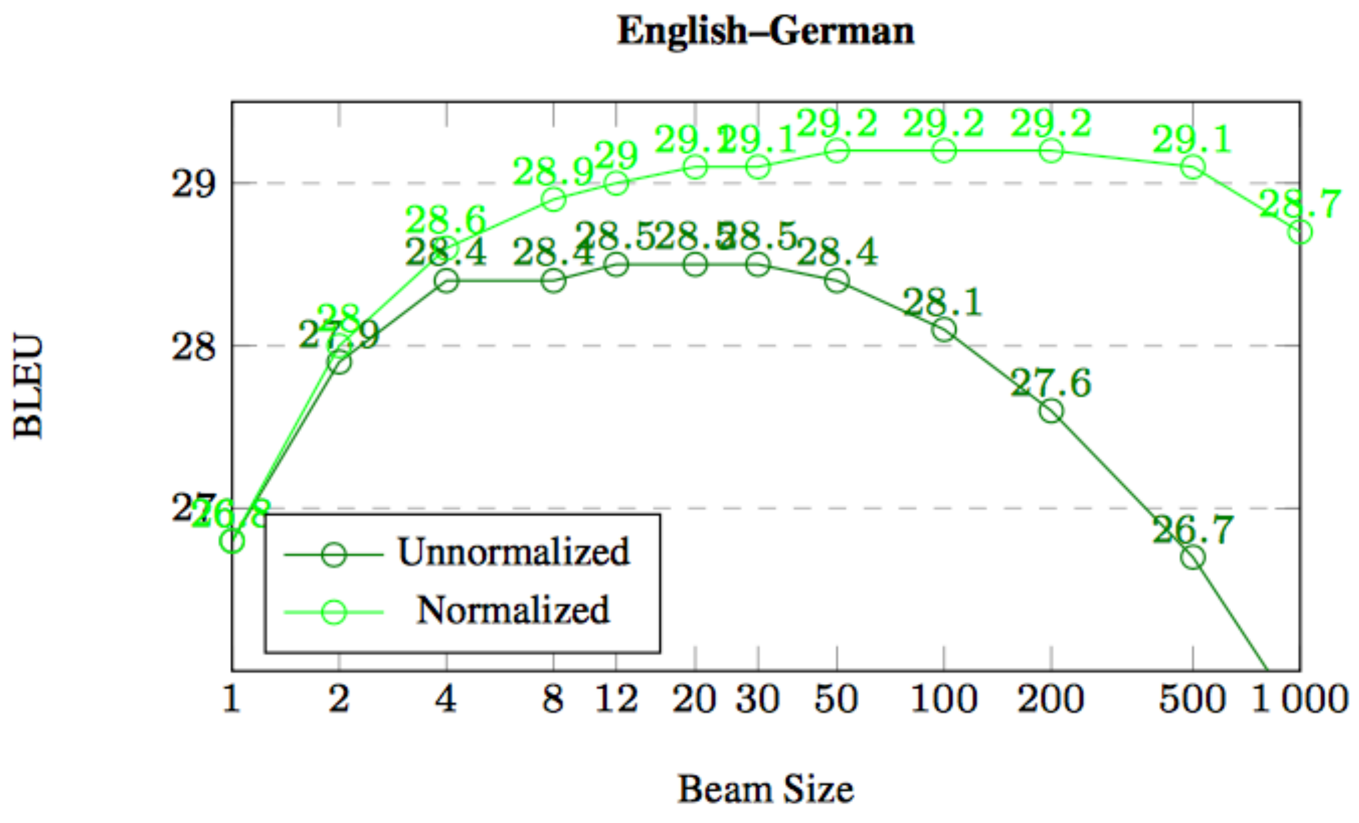
Length Norm

- Simply normalize the model score (log-prob) by the generation length:

$$S(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|\mathbf{y}|} \log p(y_i \mid \mathbf{x},\, y_{1..\{i-1\}})$$

$$\hat{S}_{\text{length\_norm}}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) / |\mathbf{y}|$$





Approaches

- To balance out log-prob's preference for short sequences, we need to estimate the proper generation length.
    - We could use the average tgt/src length ratio from the training set (Liang et al. 2017).
    - We could also train a MLP to estimate target length based off source hidden states (in this work).

Approaches

- Bounded word-reward
  - Add a tuned reward to each word up to the predicted length

$$L^*(\mathbf{x}, \mathbf{y}) = \min\{|\mathbf{y}|, L_{pred}(\mathbf{x})\}$$
$$\hat{S}_{\mathrm{BWR}^*}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) + r \cdot L^*(\mathbf{x}, \mathbf{y})$$

Approaches

- Bounded adaptive-reward
    - Add an adaptive reward based off local beam information up to the predicted length

$$\hat{S}_{\mathrm{AdaR}}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) + \textstyle\sum_{t=1}^{L^*} r_t$$

$$r_t = -(1/b) \textstyle\sum_{i=1}^{b} \log p(\mathrm{word}_i)$$

Approaches

- Brevity-penalty normalization
    - Add a brevity penalty term to length norm.

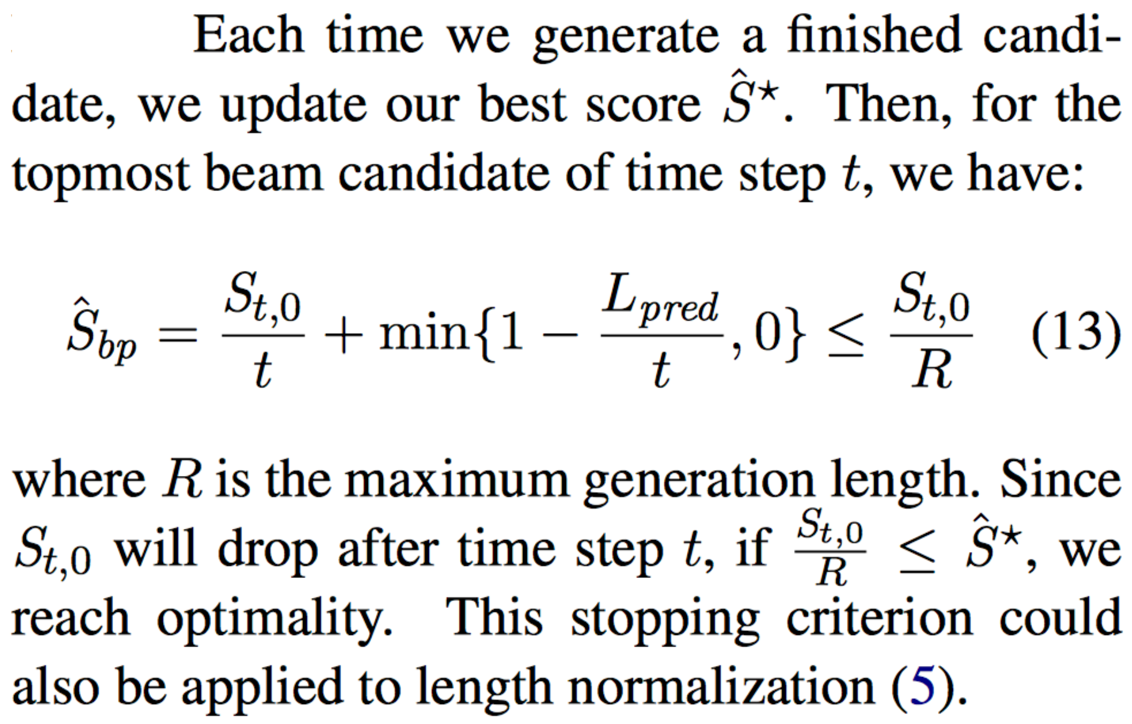
$$\hat{S}_{bp}(\mathbf{x}, \mathbf{y}) = \log bp + S(\mathbf{x}, \mathbf{y})/|\mathbf{y}|$$

where $bp = \min\{e^{1-1/lr}, 1\}$

$$lr = |\mathbf{y}|/|\mathbf{y}^*|$$

Optimal Stopping

- Provable optimal stopping exists in all above methods, disregarding loose or tight. They are all in a similiar form which proves latter candidates from beam search won't have a better (revised) model score. e.g. BP-Norm:

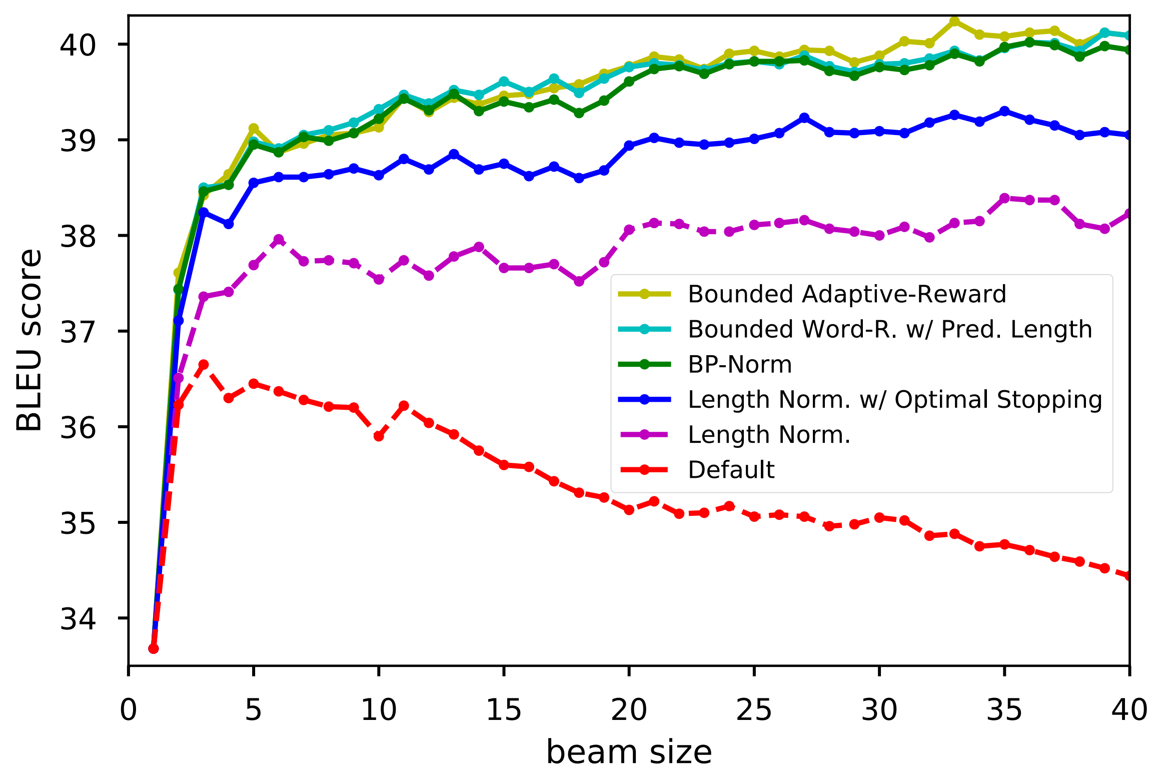


Each time we generate a finished candidate, we update our best score $\hat{S}^{\star}$. Then, for the topmost beam candidate of time step t, we have:

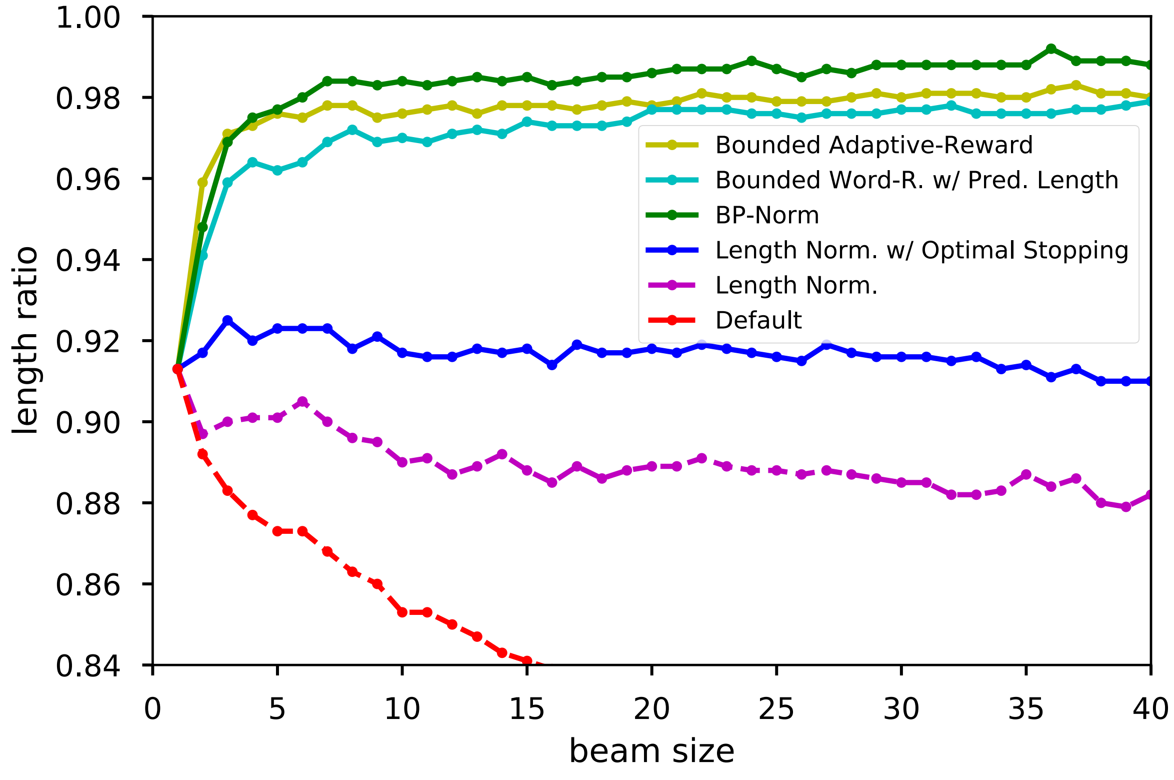
$$\hat{S}_{bp} = \frac{S_{t,0}}{t} + \min\{1 - \frac{L_{pred}}{t}, 0\} \leq \frac{S_{t,0}}{R} \quad (13)$$

where R is the maximum generation length. Since $S_{t,0}$ will drop after time step t, if $\frac{S_{t,0}}{R} \leq \hat{S}^{\star}$, we reach optimality. This stopping criterion could also be applied to length normalization (5).

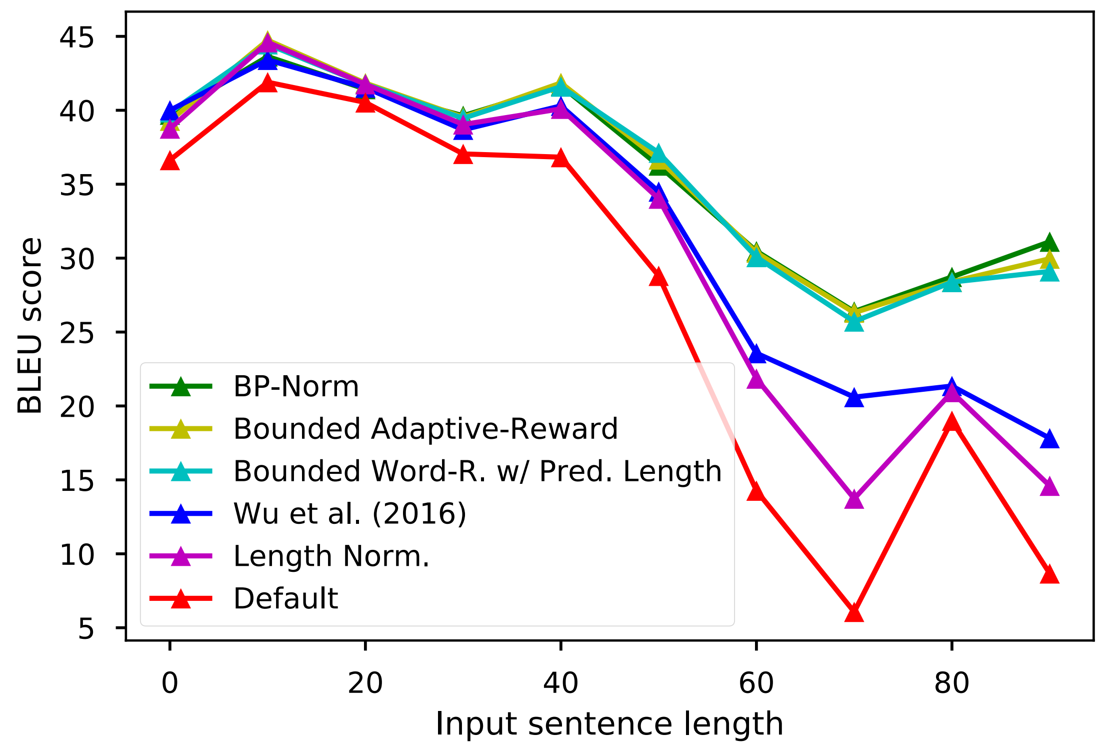
Results



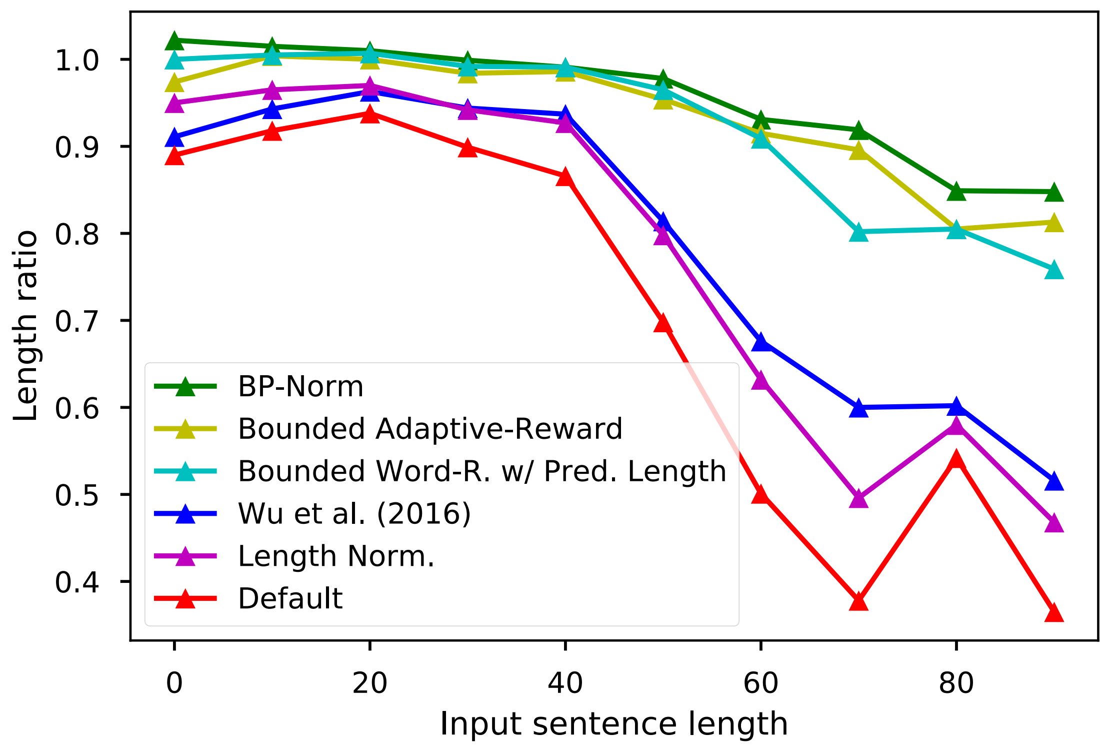
BLEU score
beam size
Bounded Adaptive-Reward
Bounded Word-R. w/ Pred. Length
BP-Norm
Length Norm. w/ Optimal Stopping
Length Norm.
Default



length ratio
beam size
Bounded Adaptive-Reward
Bounded Word-R. w/ Pred. Length
BP-Norm
Length Norm. w/ Optimal Stopping
Length Norm.
Default



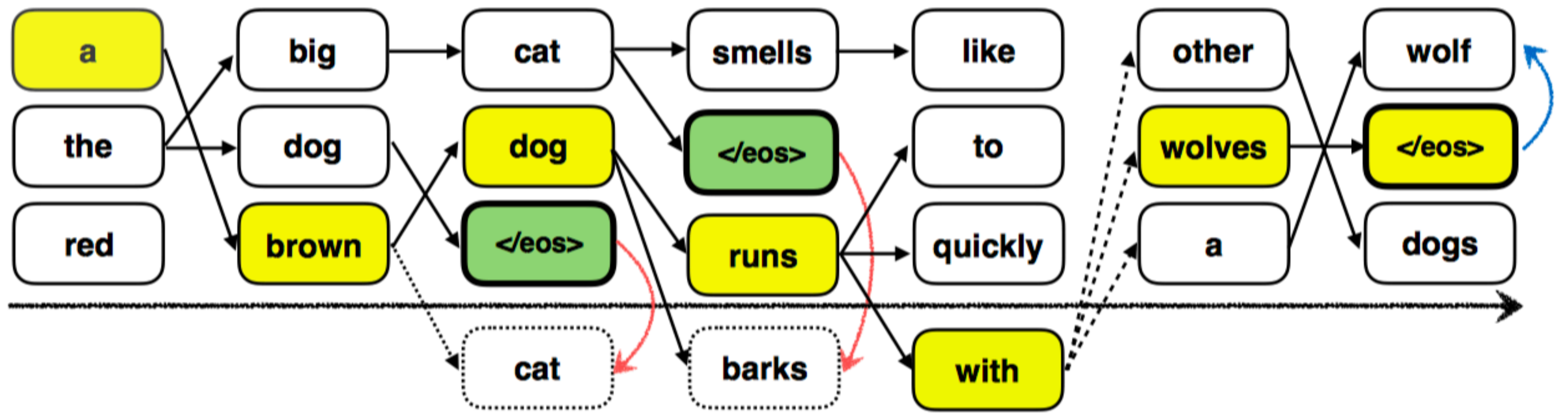
BLEU score
Input sentence length
BP-Norm
Bounded Adaptive-Reward
Bounded Word-R. w/ Pred. Length
Wu et al. (2016)
Length Norm.
Default



Length ratio
Input sentence length
BP-Norm
Bounded Adaptive-Reward
Bounded Word-R. w/ Pred. Length
Wu et al. (2016)
Length Norm.
Default

Beam Search on Training

- Ma et al. (2019) is largely based on another work (BSO), which bring beam search into training, and penalize whenever gold sequence falls off beam. This work adds another loss (early stop penalty) to BSO, which would penalize </eos> within the beam.



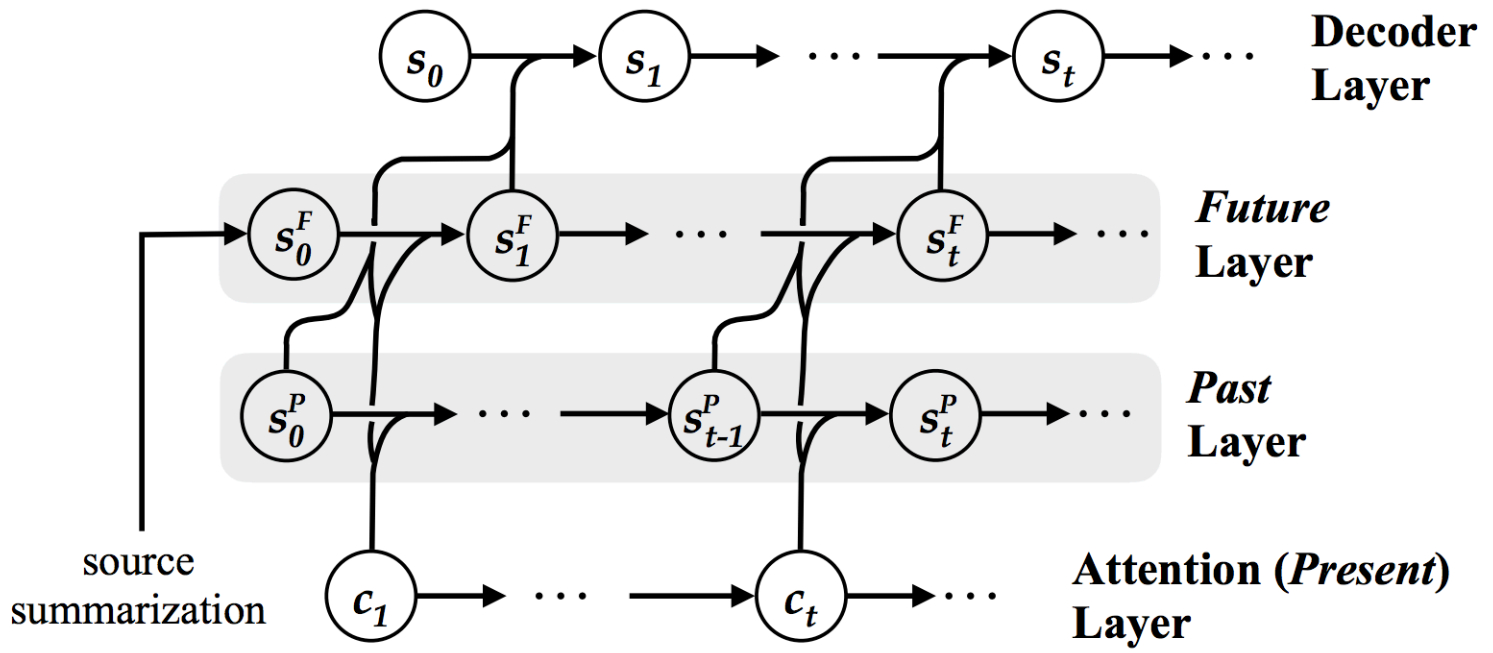
$$\mathbb{L}^s = \sum_{t=1}^{|\mathbf{y}|} \sum_{j=1}^{b} \mathbb{1}(\hat{y}_t^j = \texttt{</eos>}) \cdot Q(\hat{y}_t^j, \hat{y}_t^{b+1}) \ ,$$

$$Q(\hat{y}_t^j, \hat{y}_t^{b+1}) = (1 + f_{\mathbf{x}}(\hat{y}_t^j \mid \hat{\mathbf{y}}_{<t}^j) - f_{\mathbf{x}}(\hat{y}_t^{b+1} \mid \hat{\mathbf{y}}_{<t}^{b+1}))^+$$

Ma, M., Zheng, R., & Huang, L. (2019). Learning to Stop in Structured Prediction for Neural Machine Translation. *arXiv preprint arXiv:1904.01032*.

Improve Decoder States

- Zheng et al. (2018) add two RNN layers to the decoder, modeling past and future respectively with additional auxiliary losses. *Future loss* forces the step-wise hidden state “subtraction” to be similiar to the generated target word embedding of that step, while *past loss* modeling the “addition” part.



$$\mathbf{\Delta}_t^F = \mathbf{s}_{t-1}^F - \mathbf{s}_t^F \approx \mathbf{c}_t \approx \mathrm{E}(y_t)$$

$$loss(\mathbf{\Delta}_t^F, \mathrm{E}(y_t)) = -\log \frac{\exp\left(l(\mathbf{\Delta}_t^F, \mathrm{E}(y_t))\right)}{\sum_y \exp\left(l(\mathbf{\Delta}_t^F, \mathrm{E}(y))\right)}$$

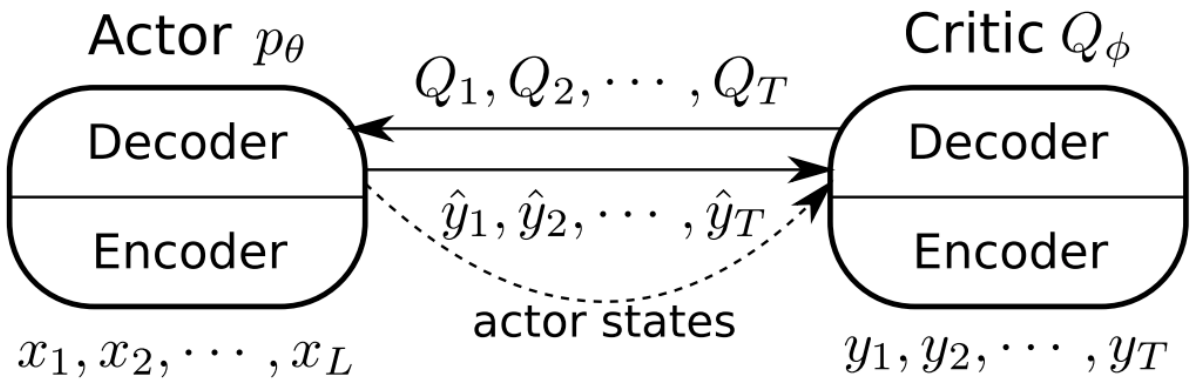
$$l(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top W \mathbf{v} + b.$$

$$\hat{\theta} = \arg\min_\theta \sum_{n=1}^{N} \sum_{t=1}^{|\mathbf{y}|} \Big\{ \underbrace{-\log P(y_t|y_{<t}, \mathbf{x}; \theta)}_{\textit{neg. log-likelihood}} + \underbrace{loss(\mathbf{\Delta}_t^F, \mathrm{E}(y_t)|\theta)}_{\text{FUTURE } loss} + \underbrace{loss(\mathbf{\Delta}_t^P, \mathrm{E}(y_t)|\theta)}_{\text{PAST } loss} \Big\}.$$

Zheng, Z., Zhou, H., Huang, S., Mou, L., Dai, X., Chen, J., & Tu, Z. (2018). Modeling past and future for neural machine translation. *Transactions of the Association of Computational Linguistics*, *6*, 145-157.

Train NMT with a Critic

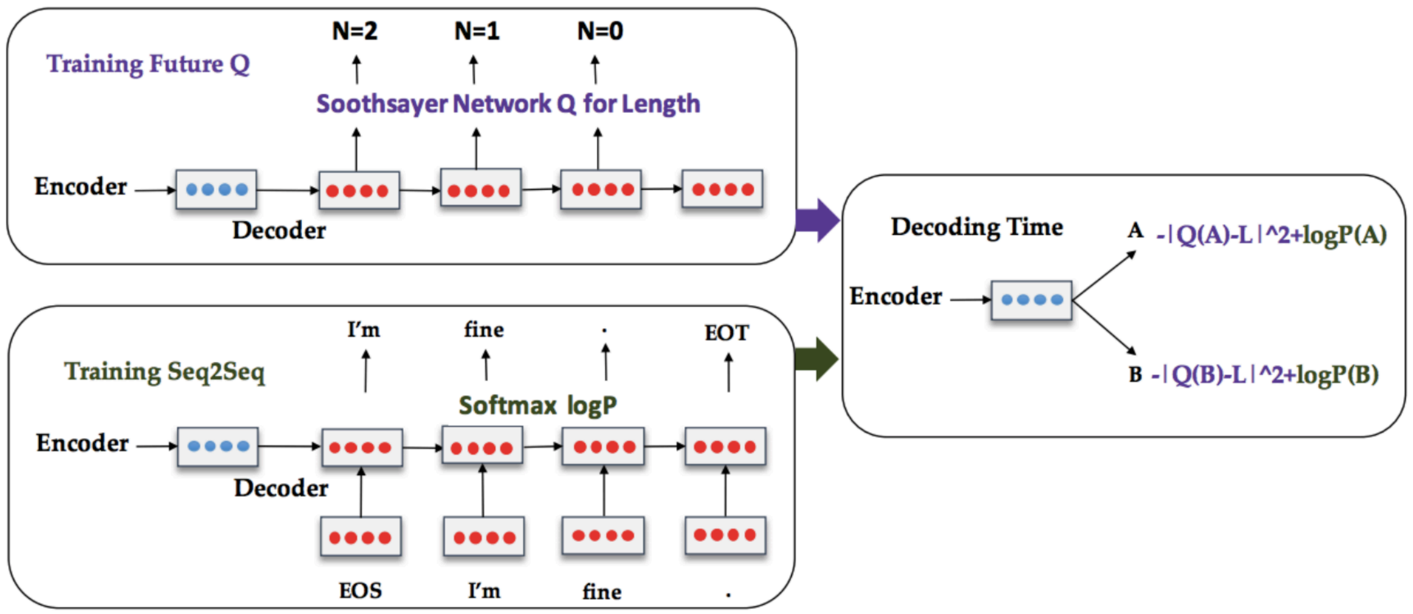
- Bahdanau et al. (2016) brings AC into NMT, and includes many practical RL tricks (delayed actor, reward shaping etc.) The most related part in this work is the *critic*, which is also a RNN defined to model the "expected future return". However, critic in this work takes golden target as input, thus can not be used in test time (decoding).



Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., ... & Bengio, Y. (2016). An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Bring Value Network into Decoding

- This work revises the ranking criterion of beam search decoding, by incorporating a Q_value with a coefficient. Q_value could takes various forms, e.g. length or BLEU.



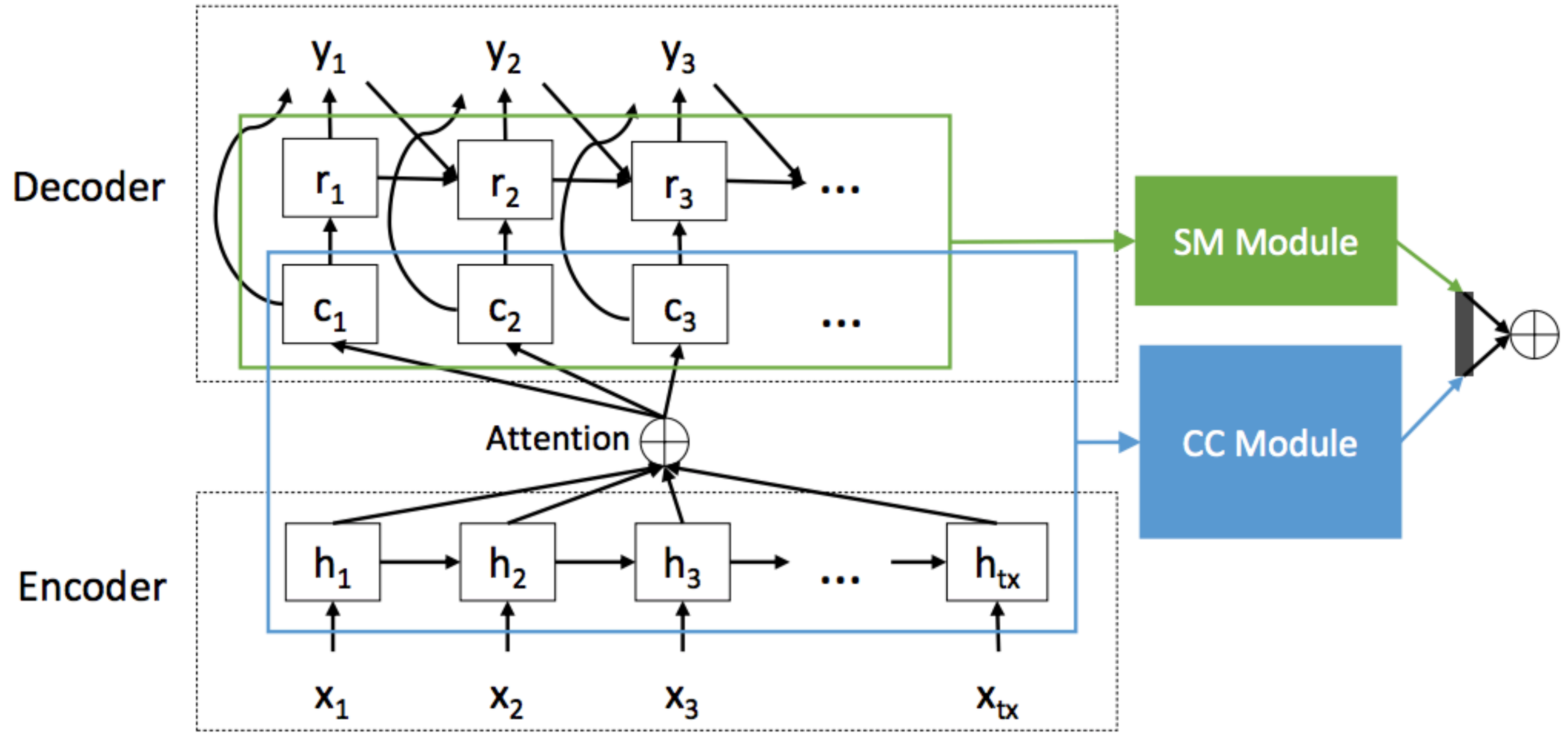
$$y_t = \arg\max_y \log p(y_{1:t}|X) - \lambda||(N-t) - Q(h_t)||^2$$

$$y_t = \arg\max_y \log p(y_{1:t-1}, y|X) + \lambda Q(y)$$

Li, J., Monroe, W., & Jurafsky, D. (2017). Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Bring Value Network into Decoding

- He et al. (2017) is very similiar to Li et al. (2017). The only differences are: 1. This work has two value network (SM, CC modules), which take difference input. 2. Instead of MSE, this work utilizes a different loss (”pairwise ranking loss”), which forces the targets with higher BLEU to be with higher Q_value.



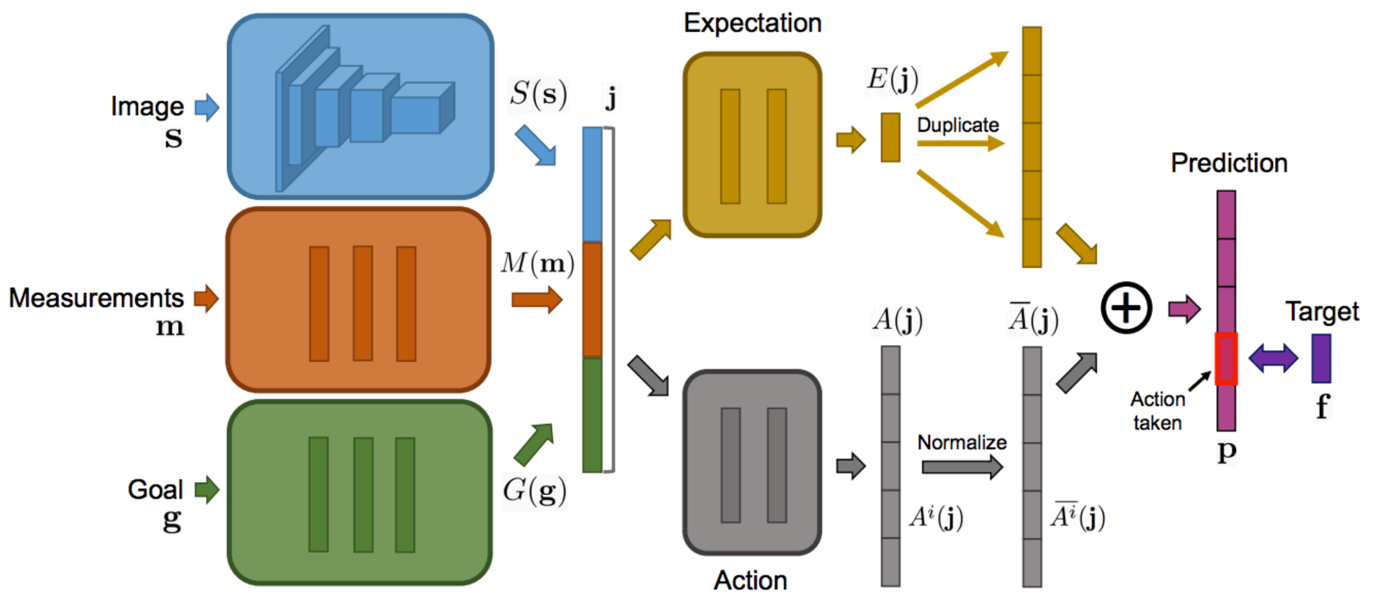
$$L(\omega) = \sum_{(x,y_{p,1},y_{p,2})} e^{v_\omega(x,y_{p,2})-v_\omega(x,y_{p,1})}$$

where $\text{avg\_bleu}(x, y_{p,1}) > \text{avg\_bleu}(x, y_{p,2})$.

He, D., Lu, H., Xia, Y., Qin, T., Wang, L., & Liu, T. Y. (2017). Decoding with value networks for neural machine translation. In *Advances in Neural Information Processing Systems* (pp. 178-187).

Densely Predict Future

- This work targets at a game called “Doom”, and presents a practical approach that surpasses SOTA RL methods (DQN, A3C). The major innovation is as follows: 1. Instead of estimating total future measurement (e.g. health), it sets the model to predict measurements 1,2,4..16 steps later, to provide richer training signals. 2. Instead of directly model the Q_value, it leverages two feedforward heads to estimate avg action values and the action values offset to the avg.



$$\overline{A^i}(\mathbf{j}) = A^i(\mathbf{j}) - \frac{1}{w}\sum_{k=1}^{w} A^k(\mathbf{j})$$

$$\mathbf{p} = \langle \mathbf{p}^{a_1}, \ldots, \mathbf{p}^{a_w} \rangle = \left\langle \overline{A^1}(\mathbf{j}) + E(\mathbf{j}), \ldots, \overline{A^w}(\mathbf{j}) + E(\mathbf{j}) \right\rangle$$

Dosovitskiy, A., & Koltun, V. (2016). Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*.