Based on OpenNMT-py, a PyTorch reimplementation of Torch-based OpenNMT (Klein et al., 2017), we used the NIST 08 newswire portion (691 sentences) for test.

**BACKGROUND: BEAM SEARCH ALGORITHM**

While decoding with a sequence-to-sequence model, we could not afford to search globally for optimal output sequence, so researchers often resort to beam search algorithm to approximate exact search. The beam search algorithm expands $B_t$ to $B_{t+1}$ as follows:

$$B_{t+1} = \text{top}\left(\left\{ (y', y, s)p(y|x, y) \mid (y', y) \in B_t \right\} \right)$$

**WHY THE CURSE EXISTS**

- As beam size increases, the more candidates it would explore. Therefore, it becomes easier to find the $<$eos$>$ symbol and terminate. Left figure shows that the $<$eos$>$ indices decrease steadily with wider beams.
- Then, because of the internal property of log-probability, shorter candidates have clear advantages over model score.
- As a conclusion, the search algorithm would find shorter candidates, and prefer even shorter ones among them.

**HOW TO BREAK THE CURSE**

**Previous Methods**
- Length Normalization (Bahdanau et al., 2014): normalize the score by its length.
- Word-Reward (He et al., 2016): add reward $r$ to each word.
- Bounded Word-Reward (Liang et al., 2017): add reward $r$ to each word up to a bound.
- Bounded Word-Reward w/ Predicted Length: We use a 2-layer MLP, which takes the mean of source hidden states as input, to predict the generation ratio $g(x)$. Then we can get our predicted length $L_{\text{pred}}(x) = g(x) | x |$.
- Bounded Word-Reward w/ Predicted Length: To favor longer generation, we add rewards $r$ to each word up to its predicted length.

$$L(x, y) = \min(\left\{ |y|, L_{\text{pred}}(x) \right\}) \quad \delta(x, y) = \alpha(x, y) + r \cdot L(x, y)$$

where $\alpha(x, y)$ is the original model score (log-probability).

**Bounded Adaptive-Reward**: Instead of a tuned reward $r$, we add an adaptive reward to each step based off local beam information. With beam size $b$, the reward for time step $t$ is the average negative log-probability of the words in the current beam.

$$r_t = \frac{1}{b} \sum_{y' \in B_t} \log p(\text{word}) \quad \delta(x, y) = \alpha(x, y) + \frac{1}{b} \sum_{y' \in B_t} r_t$$

**BP-Norm**: Instead of adding rewards, we apply brevity penalty to the length-normalized model score.

$$bp = \min(x^{1/b}, 1) \quad \delta(x, y) = \log bp + \frac{\alpha(x, y)}{|y|}$$

**DISCUSSION**

Among all methods, we recommend BP-Norm for the following reasons:
- BP-Norm works equally well with others, while doesn’t contain any hyper-parameters.
- BP-Norm is intuitive and in the same form as BLEU. Both of their exponential forms are products of brevity penalty term and geometric mean of probabilities (BP-Norm) or accuracies (BLEU).

![Figure 1: Examples of beam search algorithm with beam size 3. Red arrows denote greedy search (beam size 1).](image1)

![Figure 2: While the BLEU score drops with an increasing beam size (after 5), the brevity penalty drops with a similar curve.](image2)

![Figure 3: Left: Searching algorithm with wider beams generates $<$eos$>$ earlier. Right: The model score (log-probability) strongly prefers shorter candidates.](image3)

![Figure 4: BLEU and length ratios of various rescoring methods.](image4)

![Figure 5: BLEU and length ratios over various input sentence lengths.](image5)