**To Reviewer #1:** Choosing any two distinct scalars on the line is equivalent. Specifically, the minimizer in Proposition 1 will not change if another two distinct scalers are chosen. Therefore, we choose 0, 1 to simplify the discussion.

We experimented on using Sigmoid functions, and it does not work. We approximate the indicator vector \( A' \) by \( s(x_i - x_n) \), where \( i = 1, \ldots, n \), and \( x_n \) is the \( k \)-th smallest input. We first take \( s(\cdot) \) as a standard Sigmoid function, and it quickly runs into numerical issues even with extensive parameter tuning. We then try a hard Sigmoid function (arXiv:1811.03378) with different slopes. The best performance is 27\% on CIFAR10 (worse than a simple kNN). The gradient computed cannot provide effective guidance for the parameter updates.

**To Reviewer #2:** For beam search, \( n \) is very large. NeuralSort and Cuturi (2019) require \( O(n^2) \) memory, which is not affordable. We tried Softmax for \( k \) times (which is also proposed by us). The performance is comparable to SOFT, but it often runs into numerical issues when computing the gradient since it nests Softmax for many times. The BLEU score for a hard Top-\( k \) attention is 37.02 (SOFT 37.30). We will add more discussions in the next version.

**To Reviewer #3:** Our algorithm scales linearly w.r.t. the input size, which is not larger than any other algorithms. Furthermore, we already applied our method to a structured prediction task, i.e., machine translation, where we use beam search to search over all combinations of the tokens.

The proposed beam search method is a principled way to close the exposure bias between the training and the inference, originating from curriculum learning (Bengio, 2015, arXiv:1506.03099). Traditionally, in training, the input to the decoder is the gold sequence, while in the inference, its input is sequences decoded by beam search. So we incorporate beam search into training to close this gap.

Furthermore, the proposed method achieves a significant improvement over a very strong baseline (Bahdanau, 2014) with nearly identical hyperparameters, suggesting the proposed method is more than tricks. (Note that BLEU of our implemented Bahdanau is 35.38, which is as far as we know the best score for RNN-based seq2seq single models. The original Bahdanau is only 28.45).

We did not consider ties because this rarely happens in practice – the input of SOFT should be float number with at least \( 10^{-5} \) precision. Furthermore, if there are ties, it does not matter which to choose among the ties for machine learning algorithms. We also remark that our method can apply to cases with ties naturally: Instead of returning a smoothed indicator vector, the entries for two tied scalars will be approximately 0.5.

**To Reviewer #5:** We will include more discussions on the sensitivity and motivation in the next version.

W1 a) For beam search, we use 100 inner iterations for a relatively large \( \epsilon (0.05) \). We use 2000 inner iterations for kNN, since we adopt a very small \( \epsilon (10^{-5}) \) for CIFAR10. For very small \( \epsilon (10^{-5}) \), we do observe significant performance drop with fewer Sinkhorn iterations. For larger \( \epsilon \), the performance is not sensitive to the number of iterations.

b) The sensitivity result of CIFAR 10 for template sample size is as follows:

<table>
<thead>
<tr>
<th>Template batch size</th>
<th>100 (current)</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. change</td>
<td>0</td>
<td>+0.40</td>
<td>+0.34</td>
<td>+0.43</td>
<td>−0.18</td>
</tr>
</tbody>
</table>

The bias introduced by small template batch size does not significantly affect the final performance.

c) Current teacher forcing ratio \( \rho \) is 0.8. With \( \rho = 0.7 \), the BLUE score will decrease 0.25. Currently we have \( \epsilon = 0.05 \), with \( \epsilon = 0.1 \), BLUE score will decrease 0.13.

d) We pad the shorter sentence (the decoded and the gold sequence) with <EOS> before feeding them into NLL.