Improving Beam Search by Removing Monotonic Constraint for Neural Machine Translation

Raphael Shu
The University of Tokyo
shu@nlab.ci.i.u-tokyo.ac.jp

Hideki Nakayama
The University of Tokyo
nakayama@ci.i.u-tokyo.ac.jp

Abstract

To achieve high translation performance, neural machine translation models usually rely on the beam search algorithm for decoding sentences. The beam search finds good candidate translations by considering multiple hypotheses of translations simultaneously. However, as the algorithm searches in a monotonic left-to-right order, a hypothesis can not be revisited once it is discarded. We found such monotonicity forces the algorithm to sacrifice some decoding paths to explore new paths. As a result, the overall quality of the hypotheses selected by the algorithm is lower than expected. To mitigate this problem, we relax the monotonic constraint of the beam search by maintaining all found hypotheses in a single priority queue and using a universal score function for hypothesis selection. The proposed algorithm allows discarded hypotheses to be recovered in a later step. Despite its simplicity, we show that the proposed decoding algorithm enhances the quality of selected hypotheses and improve the translations even for high-performance models in English-Japanese translation task.

1 Introduction

Machine translation models composed of end-to-end neural networks (Sutskever et al., 2014; Bahdanau et al., 2014; Shazeer et al., 2017; Gehring et al., 2017) are starting to become mainstream. Essentially, neural machine translation (NMT) models define a probabilistic distribution $p(y_t|y_1, ..., y_{t-1}, X)$ to generate translations. During translation phase, new words are sampled from this distribution.

As the search space of possible outputs is incredibly large, we can only afford to explore a limited number of search paths. In practice, NMT models use the beam search algorithm to generate output sequences in a limited time budget (Graves, 2012; Sutskever et al., 2014). Beam search limits the search space by considering only a fixed number of hypotheses (i.e., partial translations) in each step, and predicting next output words only for the selected hypotheses. The fixed number B is referred to as *beam size*. Beam search keeps decoding until B finished translations that end with an end-of-sequence token "$\langle/\mathrm{s}\rangle$" are found.

Comparing to the greedy search that only considers the best hypothesis in each step, beam search can find a good candidate translation that suffers in a middle step. Generally, using beam search can improve the quality of outputs over the greedy search. However, we found that the hard restriction of hypothesis selection imposed by the beam search affects the quality of the decoding paths negatively.

We can think the decoding process of an NMT model as solving a pathfinding problem, where we search for an optimal path starts from "$\langle\mathrm{s}\rangle$" and ends at "$\langle/\mathrm{s}\rangle$". For any pathfinding algorithm, a certain amount of exploration is crucial for making sure that the algorithm is following a right path. For beam search, since the beam size is fixed, it must give up some currently searching paths in order to explore new paths. The problem has a similar flavor as the exploration-exploitation dilemma in reinforcement learning. As the beam search decodes in left-to-right order monotonically, a discarded decoding path can not be recovered later.

As the decoding algorithm is essentially driven by a language model, an output with high probability (local score) is not guaranteed to have high scores for future predictions. Beam search can be trapped by such a high-confidence output. This is-

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 339–344
Melbourne, Australia, July 15 - 20, 2018.

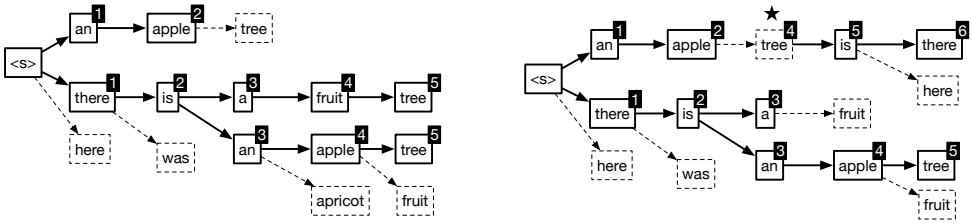

Figure 1: An intuitive comparison between beam search and single-queue decoding (SQD) with a beam size of 2. In each step, two selected hypotheses (solid boxes) and one immediately discarded hypothesis (dashed boxes) are shown in the figure. In the top right of selected hypotheses, the step numbers when they are selected are marked. The hypothesis "*an apple tree*" is discarded in step 3 in both algorithms. Comparing to beam search, SQD is able to recover this hypothesis in step 4 when other hypotheses have worse scores.

sue is more severe for language pairs that are not well aligned. One solution is to predict the expected future scores, which is considerably difficult. Another workaround for this problem is to enable the algorithm to revisit a previous hypothesis when the quality of current ones degrades.

In this work, we extend the beam search to introduce more flexibility. We manage all found hypotheses in a single priority queue so that they can be selected later when necessary. Based on a universal score function, the hypotheses with highest scores are selected to be expanded. The proposed algorithm is referred to as *single-queue decoding* (SQD) in this paper.

As the priority queue can contain massive hypotheses, we design two auxiliary score functions to help the algorithm select proper candidates. Experiments show that the proposed algorithm is able to improve the quality of selected hypotheses and thus results in better performance in English-Japanese translation task.

2 Related Work

To improve the quality of the score function in beam search, Wiseman and Rush (2016) propose to run beam search in the forward pass of training, then apply a new objective function to ensure the gold output does not fall outside the beam. An alternative approach is to correct the scores with reinforcement learning (Li et al., 2017). Diverse decoding (Li et al., 2016; Li and Jurafsky, 2016) modifies the score function for increase the diversity of hypotheses. In contrast, this work focuses on removing the constraint of beam search rather than improving the score function.

Hu et al. (2015) also describes a priority queue integrated with the standard beam search but has a different mechanism and purpose. The priority queue in their work contains top-1 hypotheses from different hypothesis stacks. In each step, only one hypothesis from the queue is allowed to be considered. Their purpose is to use the priority queue to speed up beam search at the cost of slight performance loss, which is different to this work.

As the proposed algorithm is a best-first searching algorithm, which has a flavor similar to A* search (Hart et al., 1968). Typical implementations of A* search also use a priority queue (heap) to maintain visited paths.

3 Deficiency of Beam Search

Since the beam size is fixed, when the algorithm attempts to explore multiple new decoding paths for a hypothesis, it has to discard some existing decoding paths. However, the new decoding paths may lead to bad hypotheses in the near future. As past hypotheses can not be revisited again, the beam search has to decode the hypotheses with degraded qualities continually. This phenomenon is illustrated in Fig. 1 (a), where the graph depicts the decoding process of a sentence. The correct output is supposed to be "*an apple tree is there*" or "*there is an apple tree*". In step 3, as the algorithm explores two new hypotheses in the bottom branch, the hypothesis "*an apple tree*" is discarded. In the next step, it realized that the hypothesis "*there is a*" leads to a wrong path. However, as the algorithm can not return to a discarded hypothesis, the beam search has to keep searching in the hopeless path. In this case, the candidate "*an apple tree is there*" can never be reached.

Algorithm 1 Single-queue decoding

Initialize:
  $B \leftarrow$ beam size
  $H \leftarrow$ empty hypothesis queue
  $T \leftarrow$ max steps
for $t \leftarrow 1$ to T **do**
  $S \leftarrow$ pop best B unfinished hyps from H
  $S' \leftarrow$ expand S to get $B \times K$ new hyps
  Evaluate scores of hyps in S' with Eq. 1
  Push S' into H
  if #(finished hyps in H) $\geq B$ **then**
    break
$\hat{y} \leftarrow$ best finished hyp in H
output $\hat{y}$

4 Single-Queue Decoding

In this section, we introduce an extended decoding algorithm of the beam search, which maintains a single priority queue that contains all visited hypotheses. In contrast to the standard beam search, which only considers hypotheses with the same length in each step, the proposed algorithm selected arbitrary hypotheses from the queue that may differ in length. Therefore, a hypothesis discarded in one step can be recovered in a later step.

An intuitive illustration of the proposed algorithm can be found in Fig. 1 (b). In step 4, the proposed algorithm is able to recover the hypothesis "*an apple tree*" from the queue which is discarded a previous step.

The pseudo code of the *single-queue decoding* algorithm is given in Alg. 1. Let B be the beam size. The algorithm will keep decoding until finding B finished translation candidates. The proposed decoding algorithm relies on a universal score function $\mathrm{score}(y)$ to evaluate a hypothesis y. In each step, the hypotheses with highest scores are removed from the queue to predict next words for them. New expanded hypotheses are pushed back into the queue after scoring.

In hypothesis expansion, we collect $B \times K$ (but not $B \times |V|$) hypotheses that have highest local scores (word probability). This simple filtering is essential to avoid spending to much time computing the score function. In practice, we set $K = B$.

4.1 Universal Score Function

In the proposed algorithm, a score function is required to fairly evaluate hypotheses with different lengths, which has the following form:

$$\mathrm{score}(y) = \frac{1}{|y|^{\lambda}} \log p(y|X) + \alpha\, \mathrm{PG}(y) + \beta\, \mathrm{LMP}(y). \tag{1}$$

The first part of the equation is the log probability with length-normalization, where λ is a hyperparameter that is similar to the definition of length penalty in Wu et al. (2016). We found simply utilizing this score function will sometimes cause the algorithm decode for infinite steps. To help the algorithm select proper candidates from the large queue, we designed two auxiliary penalties.

Progress Penalty: The second part of Eq. 1 is a progress penalty, which encourages the algorithm to select longer hypotheses:

$$\mathrm{PG}(y) = \begin{cases} 0 & \text{if } y \text{ finished} \\ \frac{|y|^{\gamma}}{|X|^{\gamma}} & \text{otherwise} \end{cases} \tag{2}$$

where γ are is weight that control the strength of this function. The progress penalty encourages the algorithm to select longer hypotheses.

Length Matching Penalty: The last part of Eq. 1 is a *length matching penalty*. It filters out the hypotheses that tend to produce a final translation much shorter or longer than expected.

To achieve this, we predict two Gaussian distributions. One distribution p_x^l predicts the expected length of a correct translation. Another Gaussian p_y^l predicts the expected final length if decoding a particular hypothesis. The parameters of the distributions (μ_x, σ_x and μ_y, σ_y) are predicted by an additional simple neural network, which is trained separately from the NMT model. Then we compute the cross-entropy of the two Gaussians to measure whether the expected length of translation tends to match the correct length as:

$$\mathrm{H}(p_x^l, p_y^l) = \frac{1}{2}\log(2\pi{\sigma_y}^2) + \frac{{\sigma_x}^2 + (\mu_y - \mu_x)^2}{2{\sigma_y}^2}. \tag{3}$$

We penalize a hypothesis if the cross entropy exceeds a threshold τ as:

$$\mathrm{LMP}(y) = \begin{cases} 0 & \text{if } y \text{ finished} \\ \mathbf{I}(\mathrm{H}(p_x^l, p_y^l) > \tau) & \text{otherwise} \end{cases} \tag{4}$$

where $\mathrm{I}(\cdot)$ is an indicator function.

5 Experiments

5.1 Experimental Settings

We evaluate the proposed decoding algorithm mainly with an off-the-shelf NMT model (Bahdanau et al., 2014), which has a bi-directional LSTM encoder and a single-layer LSTM decoder. The embeddings and LSTM layers have a size of 1000. We evaluate the algorithms on ASPEC English-Japanese translation task (Nakazawa et al., 2016). The vocabulary contains 80k words for English side and 40k words for the Japanese side. We report BLEU score based on a standard post-processing procedure [1].

All NMT models in this work are trained with Nesterov's accelerated gradient (Nesterov, 1983) with an initial learning rate of 0.25. The learning rate is decreased by a factor of 10 if no improvement is observed in 20k iterations. The training ends after the learning rate is annealed for three times. The models are trained on 4 GPUs; each GPU computes the gradient of a mini-batch. The gradients are averaged and distributed to each GPU using the nccl framework.

The hyperparameters of the decoding algorithms are tuned by Bayesian optimization (Snoek et al., 2012) on a small validation set composed of 500 sentences. We utilize the "bayes_opt" package for Bayesian optimization. We use the default acquisition function "ucb" with a κ value of 5. We first explore 50 initial points, then optimize for another 50 iterations.

We allow the decoding algorithms to run for a maximum of 150 steps. If the algorithm fails to find a finished translation in the limited steps, an empty translation is outputted.

5.2 Main Results

The main evaluation results are shown in Table 1, which uses a beam size of 5 as such a small beam size is more useful in a production system. The results show that the proposed single-queue decoding (SQD) algorithm significantly improves the quality of translations. With the length matching penalty (LMP), SQD outperforms the beam search with length-normalization by 1.14 BLEU on the test set. Without the progress penalty (PG), the scores are much worse.

Since SQD computes B hypotheses in batch mode at each step just like beam search, the computational cost inside the loop of Alg. 1 remains the same. The factor affecting the decoding time is the actual number of time steps. To clarify that SQD does not improve the performance by significantly increasing the number of steps, we also report the average number of steps and the decoding time for translating one sentence. We can see that it is effective applying *length matching penaltiy*. However, it slows down the algorithm as extra computation is required.

[1] We use Kytea to re-tokenize results in evaluation.

	BLEU(%)		#step	time (ms)
	valid	test		
vanilla beam search	29.61	32.87	30.3	199
w/ length-normalization	37.16	34.29	30.3	208
SQD w/o PG	38.09	34.62	36.1	238
SQD w/ PG	38.50	35.03	33.8	225
SQD w/ PG, LMP	**38.93**	**35.43**	35.0	260

Table 1: Evaluation results using a baseline model with a beam size of 5

	Test BLEU(%)			
	BS=3	BS=5	BS=8	BS=12
beam search w/ LN	37.69	37.93	38.26	38.38
SQD w/ PG	38.18	38.68	**38.98**	**39.02**
SQD w/ PG, LMP	**38.37**	**38.73**	38.89	38.98

Table 2: Evaluation results using a large NMT model with different beam sizes (BS). The scores of the beam search with length-normalization (LN) are reported as the baselines.

5.3 Experiments with a Large NMT Model

In order to see whether the performance gain can be generalized to deeper models, we train a large NMT model with two layers of LSTM decoders. We apply residual connection (He et al., 2016) to the decoder states. Before the softmax layer, an additional fully-connected layer with 600 hidden units is applied. For the attention mechanism, we use a variant of the key-value attention (Miller et al., 2016), where the keys are computed by a linear transformation of the encoder states, the queries of the attention are the sum of the feedback word embeddings and the LSTM states of the first decoder. Dropout (Srivastava et al., 2014) is applied everywhere after non-recurrent layers with a dropping rate of 0.2. To further enhance the model performance, we use byte pair encoding (Sennrich

et al., 2016) with a coding size of 40k to segment the sentences of the training data into subwords. The experiment results are shown in Table 2.

By applying various techniques, the NMT model achieves high single-model BLEU scores. The results indicate that SQD is still effective with a high-performance NMT model. The proposed algorithm is more effective with a small beam size. For this model, the contribution of *length matching penalty* is only beneficial when the beam size is smaller than 8, which may be a side-effect of applying byte pair encoding (BPE). As it is more difficult to correctly predict the number of output tokens in sub-word level.

6 Discussion

The proposed algorithm requires a block of GPU memory for storing the states of LSTM decoders for all stored hypotheses in the priority queue. The increased memory usage does not cause a problem unless a large beam size is used.

Although all hypotheses are expected to be evaluated fairly, we found only averagely 2 discarded hypotheses are recovered when decoding each sentence. The reason is that longer hypotheses tend to have higher local scores in general, which makes it difficult for the algorithm to select a short hypothesis. As a future work, a better score function is required to fully exploit the flexibility of the proposed algorithm.

7 Conclusion

In this paper, we present a novel decoding algorithm that removes the constraint imposed by the monotonicity of beam search, so that a discarded hypothesis can be revisited in a later step.

The proposed algorithm maintains all reusable hypotheses in a single priority queue. In each step, the algorithm selects hypotheses from the queue with highest scores evaluated by a universal score function. We design two auxiliary scores to help selecting proper hypotheses from a large queue.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Number 16H05872.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Newsletter*, 37:28–29.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Xiaoguang Hu, Wei Li, Xiang Lan, Hua Wu, and Haifeng Wang. 2015. Improved beam search with constrained softmax for nmt. *Proceedings of MT Summit XV*, page 297.

Jiwei Li and Daniel Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *CoRR*, abs/1601.00372.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *CoRR*, abs/1611.08562.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *LREC*.

Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *NIPS*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.