Smart-Start Decoding for Neural Machine Translation

Jian Yang[1] [*], **Shuming Ma**[2], **Dongdong Zhang**[2], **Juncheng Wan**[3], **Zhoujun Li**[1] [†], **Ming Zhou**[2]
[1]State Key Lab of Software Development Environment, Beihang University
[2]Microsoft Research Asia
[3]Shanghai Jiao Tong University
{jiaya, lizj}@buaa.edu.cn; {shumma, dozhang, mingzhou}@microsoft.com;
junchengwan@apex.sjtu.edu.cn

Abstract

Most current neural machine translation models adopt a monotonic decoding order of either left-to-right or right-to-left. In this work, we propose a novel method that breaks up the limitation of these decoding orders, called Smart-Start decoding. More specifically, our method first predicts a median word. It starts to decode the words on the right side of the median word and then generates words on the left. We evaluate the proposed Smart-Start decoding method on three datasets. Experimental results show that the proposed method can significantly outperform strong baseline models.

1 Introduction

Neural machine translation (NMT) has made remarkable progress in recent years. There has been much progress in encoder-decoder framework, including recurrent neural models (Wu et al., 2016), convolutional models (Gehring et al., 2017) and self-attention models (Vaswani et al., 2017). Particularly, the Transformer, only relying on self-attention networks, has achieved state-of-the-art performance on different benchmarks.

Most encoder-decoder frameworks generate target translation in a completely monotonic order from left to right (L2R) or from right to left (R2L). However, monotonic generation is not always the best translation order for the machine translation task. As shown in Figure 1, "乐 (happy)" needs to leverage the future context "开朗 (lively)" to make disambiguation of the translation in English sentence, because "乐" has two meanings: "happy to do something" and "Le (person name)". In this example, the L2R baseline model produced an incorrect translation of "Le (person name)" due to unseen future context.

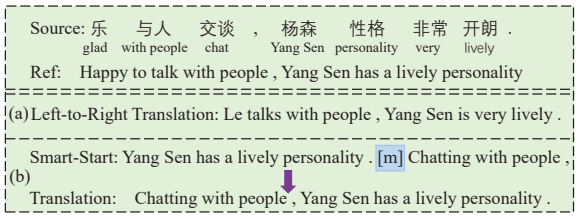

Figure 1: Example of baseline method (a) and our Smart-Start method (b). "[m]" is designed to indicate the termination of the right part generation. "[m]" is an abbreviation of "[middle]".

There are some related works on non-monotonic text generation (Mehri and Sigal, 2018; Welleck et al., 2019; Gu et al., 2019; Zhou et al., 2019b,a). Inspired by these works, we are extremely interested in considering choosing one proper position to start decoding instead of L2R or R2L order. We propose a novel method called the Smart-Start decoding method. Specifically, our method starts the generation of target words from the right part of the sentence "Yang Sen has a lively personality .", followed by the generation of the left part of the sentence "Chatting with people ,". The intuition is that humans do not always translate the sentence from the first word to the last word. Instead, humans may translate different parts of the sentence before organizing the whole translation.

As shown in Figure 1, our Smart-Start method predicts the word "Yang" in the median position of the target sentence, together with the following words of the right part of the sentence "Yang Sen has a lively personality .". Once our model produces the specific symbol "[m]" which is designed to indicate the termination of the right part generation, we will start predicting the left part of the sentence "Chatting with people ,". Finally, we obtain the final translation from the intermediate translation by solely placing the right part "Yang Sen has a lively personality ." in front of the left part and removing the additional symbol "[m]".

[*]Contribution during internship at Microsoft Research Asia.
[†]Corresponding author.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3982–3988
June 6–11, 2021.

We introduce a weighted maximum likelihood algorithm to automatically learn this kind of decoding order by giving weights to translations with different start positions.

To verify the effectiveness of our method, we conduct experiments on three benchmarks, including IWSLT14 German-English, WMT14 English-German, and LDC Chinese-English translation tasks. Experimental results show that our method outperforms monotonic and non-monotonic baselines. In conclusion, we propose a simple but effective method, which predicts from the median words to the last position's word followed by the word predictions on the left part of the sentence.

2 Smart-Start Machine Translation

In this section, we present the details of the proposed hard and soft Smart-Start methods. Our method first predicts a median word and then predicts the words on the right part, and then generates words on the left.

2.1 Method

Our method is split into two phases. First, given the source sentence $X = (x_1, x_2, \ldots, x_m)$, we use the model $P_\theta(Z_k|X)$ to predict the intermediate translation Z_k starting from the middle position of the sentence, where $Z_k = (y_{n-k+1}, \ldots, y_n, \texttt{[m]}, y_1, \ldots, y_{n-k})$ and "[m]" is the k^{th} word of Z_k. Second, we construct the final translation Y from the the intermediate translation Z_k. As shown in Figure 2, our method predicts a word y_{n-k+1}, given the source sentence. Then our model predicts the right part of sentence $(y_{n-k+1}, \ldots, y_n)$ at a time. Furthermore, when it predicts the symbol "[m]", we start predicting the left part of the sentence $(y_1, \ldots, y_{n-k})$. Then, we obtain the final translation Y from the intermediate translation Z_k. Our method is based on the Transformer architecture.

2.2 Smart-Start Decoding

Our Smart-Start method is extremely interested in breaking up the limitation of this decoding order. Different from the traditional L2R and R2L (Sennrich et al., 2016a), our Smart-Start method predicts median word y_{n-k+1} over the source sentence. Furthermore, we predict the right part of target sentence $(y_{n-k+1}, \ldots, y_n)$ sequentially which is on the right part of this word. Finally, we generate the rest words $(y_1, \ldots, y_{n-k})$ on the left part of the sentence given the source sentence and left part. Formally, we build our Smart-Start neural machine translation model as below:

$$\begin{aligned} & P_\theta(Z_k|X) \\ & = P_\theta(y_{n-k+1}|X) \times \prod_{n-k+1<i\leq n} P_\theta(y_i|X; y_k, \ldots, y_{i-1}) \\ & \times P_\theta(\texttt{[m]}|X; y_{n-k+1}, \ldots, y_n) \\ & \times \prod_{1\leq j\leq n-k} P_\theta(y_j|X; y_1, \ldots, y_{j-1}, y_{n-k+1}, \ldots, y_n) \end{aligned} \tag{1}$$

where i,j denote the i^{th} and j^{th} words in the target sentence. [m] is the k^{th} word of Z_k.

2.3 Smart-Start Training

Since there is no annotation of initial words to start the decoding, we construct the intermediate sentences with different start positions and then score them with hard or soft Smart-Start methods.

Therefore, given the source sentence X of length m and target sentence Y of length n, we can construct n intermediate sentences $Z_k = (y_{n-k+1}, \ldots, y_n, \texttt{[m]}, y_1, \ldots, y_{n-k})(k \in [1, n])$. Because the target sentence length n can be too long, we randomly sample S intermediate sentences from n intermediate sentences to construct the subset $\mathcal{S}_\mathcal{Y}$, where S is the number of sampled start positions. We apply scores calculated by the hard or soft Smart-Start methods to the loss of different intermediate samples to teach model which start position is better. This procedure can be described by the weighted log-likelihood (WML) (Dimitroff et al., 2013) reward function $\mathcal{L}$ over the dataset $\mathcal{D}$ as below:

$$\mathcal{L} = \sum_{X,Y\in\mathcal{D}} \sum_{Z_k\in\mathcal{S}_\mathcal{Y}} w_k \log P_\theta(Z_k|X) \tag{2}$$

where $\mathcal{S}_\mathcal{Y}$ is the subset containing S samples. w_k is calculated by the hard or soft Smart-Start methods.

For the hard Smart-Start method, we use the median training loss of intermediate samples as threshold to select appropriate samples to update model parameters. We calculate w_k by comparing the training loss generated by the current model of each Z_k from $\mathcal{S}_\mathcal{Y}$ with the threshold as below:

$$w_k = \delta_{\mathcal{L}_k \geq \mathcal{L}_{med}} \tag{3}$$

where $\delta_{\mathcal{L}_k \geq \mathcal{L}_{med}}$ equals to 1 if $\mathcal{L}_k \geq \mathcal{L}_{med}$ else 0. $\mathcal{L}_{med}$ is the median loss of the sample in $\mathcal{S}_\mathcal{Y}$. For each intermediate sentence $Z_k \in \mathcal{S}_\mathcal{Y}$, the objective of Z_k is denoted as $\mathcal{L}_k = \log P_\theta(Z_k|X)$.

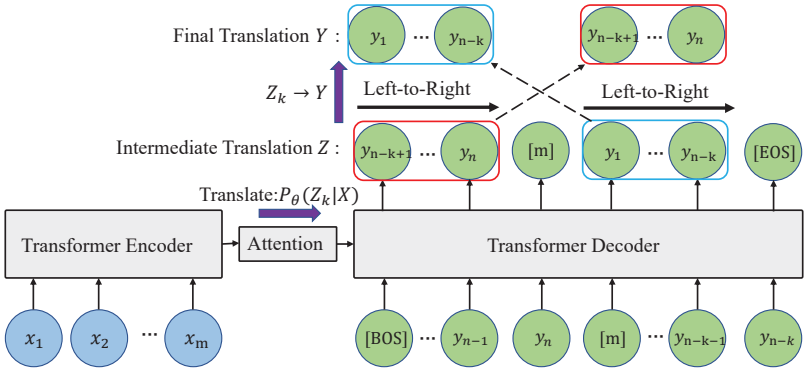

Figure 2: Overview of our Smart-Start method.

The soft Smart-Start method uses BLEU metric to evaluate intermediate samples with different start positions. It calculates BLEU points between the translation Z_k^{trans} and the reference Z_k. Softmax function is used to reweigh the w_k as below:

$$w_k = \underset{Z_k \in \mathcal{S}_\mathcal{Y}}{\text{Softmax}}(\text{BLEU}(Z_k^{trans}, Z_k)) \quad (4)$$

where Z_k^{trans} is the intermediate translation generated by the current training model $P_\theta(Z_k|X)$ using the teacher forcing method. Z_k is the intermediate sentence from $\mathcal{S}_\mathcal{Y}$.

3 Experiments

In this section, we evaluate our method on three popular benchmarks.

3.1 Dataset

IWSLT14 De-En corpus contains 16K training sequence pairs. The valid and test set both contain 7K sentence pairs. **LDC Zh-En corpus** is from the LDC corpus. The training data contains 1.4M sentence pairs. NIST 2006 is used as the valid set. NIST 2002, 2003, 2005, 2008, and 2012 are used as test sets. **WMT14 En-De corpus** has 4.5M sentence pairs. The newstest2013 and the newstest2014 are used as valid the test set. All languages are tokenized by Moses (Koehn et al., 2007) and our Chinese tokenizer, and then encoded using byte pair encoding (BPE) (Sennrich et al., 2016b) with 40K merge operations. The evaluation metric is BLEU (Papineni et al., 2002).

3.2 Training Details

We conduct experiments on 8 NVIDIA 32G V100 GPUs and set batch size as 1024 tokens. In the training stage, we adopt the Adam optimizer

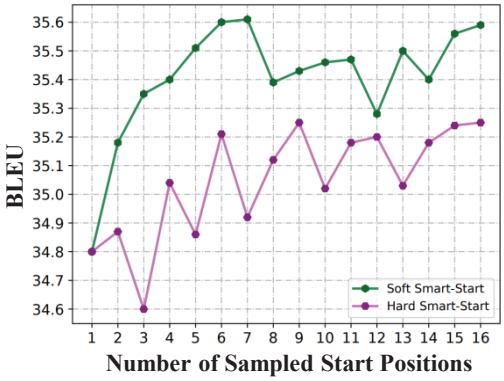

Figure 3: Results of different values of the number of sampled start positions on IWSLT14 De→En test set.

($\beta_1 = 0.9, \beta_2 = 0.98$) (Kingma and Ba, 2015) using the inverse sqrt learning rate schedule (Vaswani et al., 2017) with a learning rate of 0.1 and 4000 warming-up steps. We set the number of sampled start positions $S = 8$ described as Equation 2.

For the **LDC Zh→En translation task**, we use the *Transformer_base* setting with the embedding size as 512 and feed-forward network (FFN) size as 2048. For the **IWSLT14 De→En translation task**, we use the *Transformer_small* setting with embedding size as 512 and FFN size as 1024. The dropout is set as 0.3 and weight decay as 0.0001 to prevent overfitting. For the **WMT14 En→De translation task**, we use the *Transformer_big* setting with embedding size as 1024 and FFN size as 4096. Following the previos work (Ott et al., 2018), we accumulate the gradient for 16 iterations to simulate a 128-GPU environment.

3.3 Baselines and Results

We compare our method with the other baselines, including Transformer (Vaswani et al., 2017), RP Transformer (Shaw et al., 2018), Light-

Zh → En	MT06	MT02	MT03	MT05	MT08	MT12	Avg
LightConv (Wu et al., 2019)	43.41	42.63	45.02	43.93	35.95	34.61	40.43
DynamicConv (Wu et al., 2019)	43.65	42.60	45.80	43.60	36.91	35.55	40.89
Transformer (our implementation)	44.55	44.60	46.55	45.81	36.57	35.10	41.73
Hard Smart-Start (our method)	45.15	44.62	47.59	46.75	38.52	36.83	42.86
Soft Smart-Start (our method)	**45.70**	**44.61**	**48.10**	**47.63**	**39.18**	**37.66**	**43.44**

Table 1: Case-insensitive evaluation results on LDC Zh→En translation task with BLEU-4 scores (%). The "Avg" column means the averaged result of all NIST test sets. All baselines are re-implemented by ourselves.

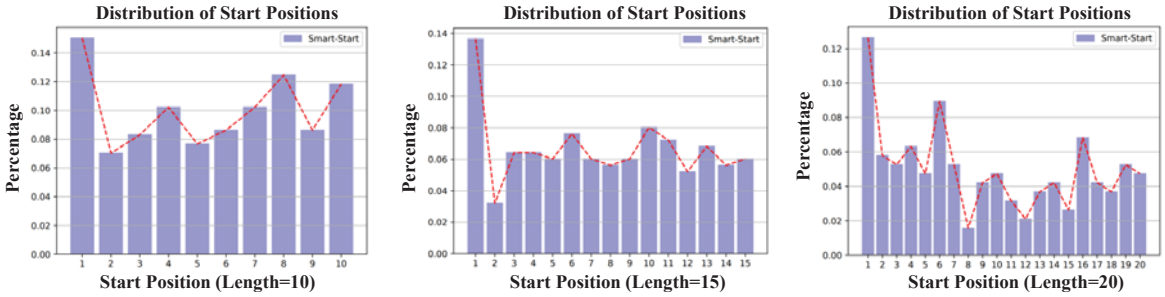

Figure 4: The distribution of different start positions of our approach. We counted the location of the start tag [m] in the intermediate translation Z. The k^{th} position represents the sentence $Z_k = (y_{n-k+1}, \ldots, y_n, \texttt{[m]}, y_1, \ldots, y_{n-k})$. Translations of our Smart-Start method with a length of 10, 15, 20 separately contain 313, 249, and 190 samples from the IWSLT14 German→English test set.

De → En	BLEU
LightConv (Wu et al., 2019)	34.80
DynamicConv (Wu et al., 2019)	35.20
Transformer (our implementation)	34.63
Hard Smart-Start (our method)	35.25
Soft Smart-Start (our method)	**35.61**

Table 2: Case-insensitive BLEU-4 scores (%) on IWSLT14 De→En translation task.

En → De	BLEU
RP Transformer (Shaw et al., 2018)	29.20
SB-NMT (Zhou et al., 2019a)	29.21
LightConv (Wu et al., 2019)	28.90
DynamicConv (Wu et al., 2019)	29.70
Transformer (our implementation)	29.36
Hard Smart-Start (our method)	29.45
Soft Smart-Start (our method)	**30.01**

Table 3: Case-sensitive BLEU-4 scores (%) on WMT14 En→De translation task.

Conv/DynamicConv (Wu et al., 2019), and SB-NMT (Zhou et al., 2019a).

For the results of IWSLT14 De→En in Table 2 and LDC Zh→En machine translation tasks in Table 1, our soft method significantly gets an improvement of +0.98/+1.71 BLEU points than a strong Transformer model.

For the WMT14 En→De task, the results of our model are presented in Table 3. Besides, we also compare our method with other self-attention models. The SB-NMT model gets a BLEU points of 29.21 which decodes from L2R and R2L simultaneously and interactively. Our method achieves an improvement of +0.56 BLEU points over the Transformer baseline. Besides, our soft Smart-Start method outperforms the SB-NMT model by +0.80 BLEU points.

3.4 Discussions and Analysis

Number of Sampled Start Positions To explore the effect of the number of sampled start positions S described as Equation 2, we conduct experiments on the IWSLT14 De→En translation task. Figure 3 shows that our hard and soft Smart-Start methods have gradually improved performance by increasing the value of S. Soft Smart-Start method outperforms the hard method under different settings. The soft method achieves a higher BLEU score when the number of sampled start positions equals 7. The proper interval ($4 \leq S \leq 12$) is recommended to use in our method. In conclusion, the

soft Smart-Start method can bring a more positive influence on BLEU scores.

Distribution of Start Positions During the inference stage, our model generates intermediate translation Z_k, where `[m]` is in the k^{th} position. We explore the distribution of the positions of symbol `[m]`. We separately collect all translations, the length of which equals 10, 15, and 20 tokens. For example, in the left picture of Figure 4, we count the positions of `[m]` in all sentences with a length of 10. Also, the middle picture reports the positions of sentences with a length of 15 and the right picture reports these sentences with a length of 20. Figure 4 shows that other positions in the sentence also occupy a certain proportion. Therefore, the conventional left-to-right decoding order is not always the best decoding order, and starting from other positions is beneficial for translation quality, which verifies our motivation.

Linguistic Analysis Based on the Figure 4, we further try making linguistic analysis. Three pictures show that the `[m]` tends to occur in the 1^{th} position, where the intermediate translation is $Z_1 = (y_n, \texttt{[m]}, y_1, \ldots, y_{n-1})$. We observe that y_n mostly is the punctuation such as period, question mark, and exclamation mark under this situation. Conjunction and preposition words are also inclined to appear at the beginning of sentences such as "or" and "but", which indicates clauses are easier to be placed at the beginning. It is consistent with our intuition that punctuation marks are most easy to predict at first.

De → En	Training time (hours)	BLEU (%)
Transformer	0.9	34.6
Our method	1.8	35.4

Table 4: The comparison of the training time and the model performance between the Transformer baseline and our method on the IWSLT14 De→En translation task. Both experiments are conducted on the 8-V100-GPU environment. To save the training time, we choose a small value of the number of sampled start positions 4 to save time in the practical scenario.

Training Time The Transformer baseline costs nearly 0.9 hours and our method costs nearly 1.8 hours (only ×2 lower speed) on the IWSLT-2014 De→En translation task, where both experiments are conducted on the 8-V100-GPU environment with 1024 max tokens. Our method doesn't require many additional training steps to converge compared with the Transformer baseline. Our method outperforms the Transformer baseline by +0.8 BLEU points. Another factor affecting the training time is the number of sampled start positions. We also investigate the proper value of the number of sampled start positions. In practice, smaller value such as 4 or 6 can also bring significant improvements. Therefore, we choose a smaller value of the sampled start positions and use multiple GPUs to keep the training time in a reasonable range.

4 Related Work

Neural Machine Translation (NMT) has attracted a lot of attention recently. The architecture of NMT models has evolved quickly so that there are many different models (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Kalchbrenner et al., 2016; Gehring et al., 2017; Vaswani et al., 2017; He et al., 2018). Asynchronous and synchronous Bidirectional decoding Model (Zhang et al., 2018; Zhou et al., 2019b) exploits the contexts generated in the R2L manner to help the L2R translation. Previous non-monotonic methods (Serdyuk et al., 2018; Zhang et al., 2018; Zhou et al., 2019a,b; Zhang et al., 2019; Welleck et al., 2019) jointly leverage L2R and R2L information. Non-monotonic methods are also widely used in many tasks (Huang et al., 2018; Shu and Nakayama, 2018), such as parsing (Goldberg and Elhadad, 2010), image caption (Mehri and Sigal, 2018), and dependency parsing (Kiperwasser and Goldberg, 2016; Li et al., 2019). Similarly, insertion-based method (Gu et al., 2019; Stern et al., 2019) predicts the next token and its position to be inserted.

5 Conclusion

In this work, we propose a novel method that breaks up the limitation of these decoding orders, called Smart-Start decoding. Our method predicts a median word and then generates the words on the right part. Finally, it generates words on the left. Experimental results show that our Smart-Start method significantly improves the quality of translation.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126), and the Fund of the State Key Laboratory of Software Development Environment(No.SKLSDE-2021ZX).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Georgi Dimitroff, Laura Tolosi, Borislav Popov, and Georgi Georgiev. 2013. Weighted maximum likelihood loss as a convenient shortcut to optimizing the f-measure of maximum entropy classifiers. In *RANLP 2013*, pages 207–214.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*, pages 1243–1252.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *NAACL 2010*, pages 742–750.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *CoRR*, abs/1902.01370.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *NeurIPS 2018*, pages 7955–7965.

Liang Huang, Kai Zhao, and Mingbo Ma. 2018. When to finish? optimal beam search for neural text generation (modulo beam size). *CoRR*, abs/1809.00069.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree lstms. *TACL*, 4:445–461.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2017*.

Zuchao Li, Jiaxun Cai, and Hai Zhao. 2019. Effective representation for easy-first dependency parsing. In *PRICAI 2019*, pages 351–363.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*, pages 1412–1421.

Shikib Mehri and Leonid Sigal. 2018. Middle-out decoding. In *NeurIPS 2018*, pages 5523–5534.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *WMT 2018*, pages 1–9.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL 2012*, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *WMT 2016*, pages 371–376.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL 2016*, pages 1715–1725.

Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Adam Trischler, Chris Pal, and Yoshua Bengio. 2018. Twin networks: Matching the future for sequence generation. In *ICLR 2018*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL 2018*, pages 464–468.

Raphael Shu and Hideki Nakayama. 2018. Improving beam search by removing monotonic constraint for neural machine translation. In *ACL 2018*, pages 339–344.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML 2019*, pages 5976–5985.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017*, pages 5998–6008.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *ICML 2019*, pages 6716–6726.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *ICLR 2019*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *AAAI 2018*, pages 5698–5705.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Tong Xu. 2019. Regularizing neural machine translation by target-bidirectional agreement. In *AAAI 2019*, pages 443–450.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019a. Synchronous bidirectional neural machine translation. *TACL*, 7:91–105.

Long Zhou, Jiajun Zhang, Chengqing Zong, and Heng Yu. 2019b. Sequence generation: From both sides to the middle. In *IJCAI 2019*, pages 5471–5477.