SDPSAT: Syntactic Dependency Parsing Structure-Guided Semi-Autoregressive Machine Translation

Xinran Chen, Yuran Zhao, Jianming Guo, Sufeng Duan, and Gongshen Liu(✉)

School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University, Shanghai, China
{jasminechen123,zyr527,jsguojianming,1140339019dsf,lgshen}@sjtu.edu.cn

Abstract. The advent of non-autoregressive machine translation (NAT) accelerates the decoding superior to autoregressive machine translation (AT) significantly, while bringing about a performance decrease. Semi-autoregressive neural machine translation (SAT), as a compromise, enjoys the merits of both autoregressive and non-autoregressive decoding. However, current SAT methods face the challenges of information-limited initialization and rigorous termination. This paper develops a layer-and-length-based syntactic labeling method and introduces a syntactic dependency parsing structure-guided two-stage semi-autoregressive translation (SDPSAT) structure, which addresses the above challenges with a syntax-based initialization and termination. Additionally, we also present a Mixed Training strategy to shrink exposure bias. Experiments on six widely-used datasets reveal that our SDPSAT surpasses traditional SAT models with reduced word repetition and achieves competitive results with the AT baseline at a $2\times \sim 3\times$ speedup.

Keywords: Non-autoregressive · Machine translation · Syntactic dependency parsing

1 Introduction

While autoregressive neural machine translation (AT) maintains cutting-edge performance, its applications in large-scale and real-time scenarios are severely restricted by the slow inference speed [3,4]. In contrast, non-autoregressive (NAT) models, based on the independence hypothesis, significantly increase inference speed through parallel decoding but experience reduced performance [4,5,10]. As a compromise between AT and NAT, semi-autoregressive (SAT) models [15,21] utilize both autoregressive and non-autoregressive properties in their decoding, in which SAT models not only capture target-side dependencies more effectively than NAT models, but also enhance translation efficiency beyond AT model [15,21,22].

The primitive equal-length segmented SAT models in [15], which decode sentence segments non-autoregressively and generate words within those segments

B. Luo et al. (Eds.): ICONIP 2023, CCIS 1962, pp. 604–616, 2024.
https://doi.org/10.1007/978-981-99-8132-8_45

autoregressively, adopt the independence assumption and encounter the multi-modality problem—the multi-modal distribution of target translations is difficult to capture [4]. To handle the multi-modality errors in primitive SAT of equal-length segmentation, [15] further proposes to expose the model to some unequal-length segmented samples in training and remove duplicate segments in inference. However, its mixed segmentation criterion is not clear enough, and its additional operation to remove repetitive segments is not straightforward. Section 5 shows detailed translation examples.

Different from [15], we attribute the multi-modality problem of SAT to two limitations: (i) **information-limited initialization:** during decoding, the SAT decoder which is initialized by a sequence of `[BOS]`, lacks instructions on the subsequent prediction. (ii) **rigorous termination:** for SAT, dividing sentences into equal-length segments is simplest and most time-efficient [15], while it would lead to SAT learning the rigorous equal-length termination pattern. The multi-modality problem leads to repeated or absent tokens. For example, in translation *"`[BOS]` There are lots of `[BOS]` of flowers outside."*, the first segment decodes the final word *"of"* to keep the same length as the second segment, which starts decoding from *"of"*, and results in repetition. These same factors also contribute to the token missing, like *"`[BOS]` There are lots `[BOS]` flowers outside."*, where *"of"* is omitted. Therefore, making initialization more informative and termination more reasonable in SAT is worth exploring.

In this work, we present a **S**yntactic **D**ependency **P**arsing structure-guided **S**emi-**A**utoregressive **T**ranslation model (**SDPSAT**) to overcome the two limitations above. In a syntactic dependency parsing tree, each branch corresponds to one sentence segment. As decoding branches from the root, corresponding sentence segments are decoded parallelly, which fits well with the global non-autoregressive and local autoregressive decoding scheme in SAT. Inspired by the decoding of syntactic dependency parsing tree and SAT, we design a layer-and-length-based tree transformation and traversal techniques to generate the syntactic structure labels. Specifically, the syntactic labels in the syntactic tree serve for two primary functions. Firstly, the syntactic structure labels are used as prediction guides, which offer content guidance for improved initialization during subsequent inferences. Secondly, syntactic structure labels also determine termination criteria in which sentences are divided into semantically consecutive segments throughout the training process. Additionally, we provide a Mixed Training technique to shrink the *exposure bias* between training and inference.

Our SDPSAT delivers the best translation quality within the SAT group and achieves a comparable translation quality with AT baselines on six benchmarks with a $2\times \sim 3\times$ speedup. Furthermore, SDPSAT achieves a low word repetition rate of about 0.30%–0.60%. The following is a summary of our contribution:

(i) We design a layer-and-length-based syntactic labeling method to integrate syntactic dependency parsing structure into SAT, which allows for a more flexible termination and a better initialization for the translation decoder.

(ii) We employ a Mixed Training strategy to mitigate the discrepancy between training and inference, ultimately enhancing translation performance.

(iii) According to experimental results on six widely-used datasets, SDPSAT not only expedites decoding but also improves translation quality with reduced repetition compared to NAT competitors.

2 Related Works

NAT accelerates machine translation inference by adopting the independence hypothesis. However, NAT faces a serious multi-modality problem [3,4], and numerous approaches have been presented to deal with it. Based on the decoding strategy, existing NAT works mainly fall into three categories: iterative NAT, fully NAT, and SAT [22]. Iterative NAT refines the translation in multiple steps at the cost of inference speed [3,5,6]. Fully NAT maintains a speed advantage by decoding in a single round and handles the multi-modality problem with the latent variables [1,10,16,27], new training objectives [8,23], and novel model architecture [26]. SAT [15,21] combines the properties of fully NAT and AT in decoding. [21] first proposes Semi-NAT with a globally AT but locally NAT decoding style, while [15] designs Semi-NAT with globally NAT but locally AT decoding methods and recovers from the repetition errors. The decoding pattern of our model aligns with that of [15].

As a fundamental technology for natural language processing (NLP), syntactic parsing analyzes the syntactic dependency between sentence components [2,7,25]. Previous researchers have introduced syntactic information to enhance NAT. [9] integrated syntactic tags at the word embedding level for NAT. [23] designed a novel training objective to eliminate the issue of syntactic multi-modality. [1] enhanced NAT by constituency parsing information. However, integrating syntactic dependency parsing into SAT is still uncharted territory. Different from the prior studies, we first present an innovative two-stage syntactic dependency parsing-based SAT framework. Our model is closely related to [1], but we differ in both aspects of modeling syntactic structure and decoding scheme: (i) they introduce constituency labels with chunk size, while we focus on dependency parsing and design a layer-and-length-based traversal to obtain syntactic labels with depth, which shrinks syntactic vocabulary and is more conducive to model learning; (ii) they adopt a single-round Mask-Predict decoding with length predetermined constituency, while we insist on Semi-NAT decoding to better comprehend the target-side dependency within segments and own a more flexible termination, which ultimately resolves the multi-modality problem.

3 Methodology

This section introduces SDPSAT in detail, including the layer-and-length-based method to get dependency labels from the syntactic dependency tree, the two-stage decoding of SDPSAT, and the Mixed Training strategy.

3.1 Layer-and-Length-Based Syntactic Labeling

We design the layer-and-length-based method to get the syntactically annotated sentence before training, and the syntactic labels are used to supervise the training of the parse decoder. Figure 1 gives an illustration of the whole process.

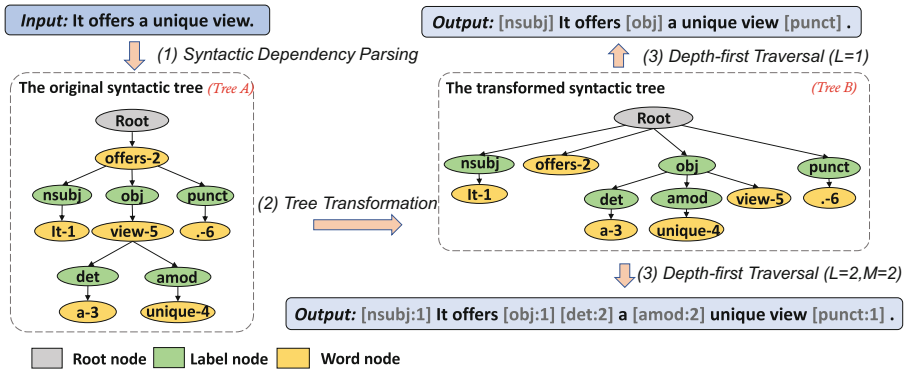

Fig. 1. Performing layer-and-length-based syntactic labeling for the input sentence. The layer and length denote the deepest dividing layer L and the maximum segment size M of the dependency parsing tree, respectively.

Dependency Parsing. Before tree transformation and traversal, we conduct syntactic dependency parsing. Each sentence is parsed into several tuples, which can be expressed formally as "*relation (governor, dependent)*", where the **governor** represents the header, the **dependent** is the modifier of the header and the **relation** represents the syntactic dependency between the header and the dependent. Inspired by [7], we treat syntactic labels (relation) and words (governor, dependent) equally as tree nodes and represent each word node with token and positional index (*i.e.*, **It-1** in Fig. 1). The syntactic parsing tree is depicted in *Tree A* of Fig. 1.

We use *stanza* toolkits [13] to parse the target sentences, which provide leading-edge syntactic dependency parsing pre-trained models for 66 languages with highly accurate performance.

Tree Transformation and Traversal. The word and label nodes alternate in the original syntactic tree branches, and make the tree structure complicated. To match our proposed two-stage SAT decoding, we simplify the tree based on the deepest dividing layer L and the maximum segment size M. The label nodes deeper than L will be ignored in traversal for simplification. If the leaves of a $L-1$ (note that $L>1$) layer label node are equal to or more than M, it would be further divided according to its subsequent L layer label nodes. Restricted by the longest sentence segment, the SAT decoding could be accelerated with a larger L and an appropriate M.

Then we detail the tree transformation and traversal process. First, we move the word nodes to leaf nodes and keep the sentence sequential information for the subsequent traversal following [7]. In particular, if the positional index of the label' child is greater than its parent, the label node is adjusted to the right side of its parent, and vice versa. Second, based on L and M, we conduct a depth-first traversal on the transformed tree to get the annotated sequence. Specifically, when $L > 1$, we append the layer number of label nodes after the label to distinguish the layer (*e.g.*, `[nsubj:1]`, `[amod:2]`). *Tree B* in Fig. 1 shows the process.

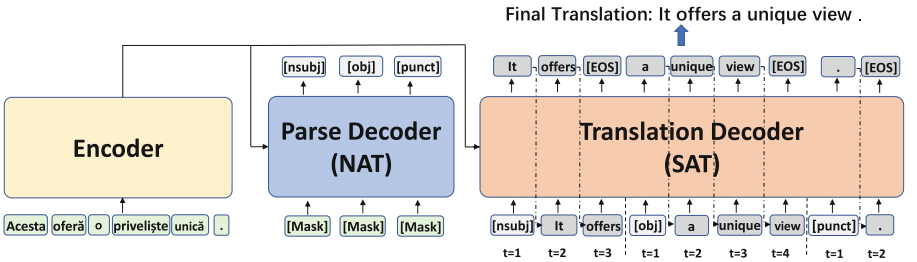

Fig. 2. An encoder and two decoders constitute SDPSAT. The translation decoder translates semi-autoregressively based on the syntactic structure given by the parse decoder. The parse decoder can be either AT or NAT. We display the NAT one here.

3.2 Two-Stage Decoding

As depicted in Fig. 2, the first SDPSAT decoding stage is to use the parse decoder to predict the syntactic structure of the target, and the second stage is to utilize the translation decoder to take the previous syntactic structure as the initialization and generate segments simultaneously.

Stage 1: Syntactic Labels Prediction. In this stage, we predict the syntactic labels of the transformed syntactic dependency parsing tree, which serve as syntactic structure hints for the next decoding stage. We design two feasible decoding schemes for this stage:

(i) Fully NAT We adopt Mask-Predict [3] to predict the label nodes of the transformed tree. Following the setting of [3], we employ a length predictor in the encoder to determine the length of the sentence before prediction. With the source sentence X, the probability distribution of the corresponding syntactic label Z is:

$$P\left(Z|X\right) = P\left(n|X\right) \prod_{t=1}^{n} P\left(z_t|X\right), \tag{1}$$

where n refers to the length of the syntactic label sequence. During training, we mask a specific percentage of tokens randomly and calculate the loss of the observed tokens, following [3]. As for inference, the parse decoder receives a sequence of `[MASK]` as input and generates the syntactic label sequence.

(ii) Fully AT A light fully-AT decoder, which only contains two layers, is also introduced to generate the syntactic label sequence, and its conditional probability could be expressed as:

$$P\left(Z|X\right) = \prod_{t=1}^{n} P\left(z_t|z_{<t}, X\right), \tag{2}$$

where $z_{<t}$ indicates the sequence generated prior to time-step t. We experimentally demonstrate that the AT decoder can better model the syntactic structure information, and its shallow decoder allows it to achieve decoding times close to those of NAT.

For simplicity, SDPSAT with an AT or a NAT parse decoder is abbreviated as SDPSAT (AT) or SDPSAT (NAT), respectively.

Stage 2: Words Prediction. With the guidance of syntactic structure labels generated by the parse decoder, the translation decoder predicts the target words semi-autoregressively, and the translation distribution could be written as:

$$P\left(Y|X\right) = \prod_{t=1}^{max\, n_i} P\left(S_t|S_{<t}, X, Z\right), \tag{3}$$

where S_t and $S_{<t}$ denote the predicted words in and before time-step t of all the segments, and Z presents the syntactic sequence. For a target sentence with segment lengths of $\{n_1, n_2, ..., n_k\}$, the total decoding step is $max\, n_i$.

3.3 Mixed Training Strategy

During training, the translation decoder is initialized with the ground truth syntactic labels, while in inference, it receives the generated syntactic labels of the parse decoder as initialization. This data distribution discrepancy, called *exposure bias* [17], may harm the translation performance.

Inspired by [24], we design a strategy called *Mixed Training* to shrink this discrepancy, which replaces the ground truth syntactic labels with predicted ones in the input of the translation decoder during training with an increasing probability. We define the probability p as follows:

$$p = \min\left\{0.5, \gamma\frac{t}{T}\right\}, \tag{4}$$

where t and T represent the current and maximum steps respectively, and γ indicates the rate controlling parameter. For better convergence, we conduct the strategy after training for 20 epochs.

For the parse decoder of AT, we adopt Force Decoding [24] to ensure the length of syntactic label prediction the same as the ground truth.

4 Experiments

4.1 Setup

Datasets Preprocess. The details of datasets preparation are as follows.

(i) **Datasets** We use the preprocessed datasets WMT14 En↔De (about 4.5M pairs), WMT16 En↔Ro (about 610k pairs) and WMT17 En↔Zh (about 20M pairs) from [3] for a fair comparison with prior studies. We use sequence-level knowledge distillation datasets for training. Specifically, the original target sentences are replaced with the translation of the standard AT Transformer to mitigate the multi-modality problem.

(ii) **Preprocess** The parallel sentences are preprocessed by Moses toolkits[1] and partitioned into subwords with Byte-Pair Encoding [18]. The target training set is processed by Stanza[2] to obtain the syntactic labels. We retrieve around 50 and 100 syntactic labels for each dataset under the conditions of $L = 1$ and $L = 2, M = 10$. The syntactic labels constitute the vocabulary of the parse decoder. The syntactic labels and words together make up the shared vocabulary of the encoder and the translation decoder.

Model Configurations. We use OpenNMT-py[3] as the framework. Our model is based on Transformer (base) [20] and we strictly follow its parameter settings. Specifically, we implement SDPSAT with a 6-layer encoder and a 6-layer translation decoder. As for the parse decoder, we adopt a 2-layer AT decoder and a 6-layer NAT decoder, respectively.

Key Baseline. AT Transformer and a series of NAT models are selected as baselines, including iterative NAT, fully NAT and Semi-NAT. Among Semi-NAT, there are two different decoding schemes, denoted as GALN and GNLA. "GNLA" is the abbreviation of **g**lobally **n**on-autoregressive but **l**ocally **a**utoregressive scheme, and "GALN" is the short for **g**lobally **a**utoregressive but **l**ocally **n**on-autoregressive scheme. GNLA-SAT divides the sentence into equal-length segments, and it is the common primitive model of RecoverSAT [15] and our SDPSAT. As the decoding paradigm of SDPSAT aligns with RecoverSAT and GNLA-SAT, we choose them as our key baselines and reimplement GNLA-SAT. GALN-SAT [21] decodes in the opposite way to ours.

4.2 Inference and Evaluation

We use the greedy search strategy for both first-stage (AT) and second-stage decoding. Following previous works, we evaluate the translation quality with BLEU [11] for all the datasets except WMT17 En→Zh, which is tested by SacreBLEU [12]. For the inference speed, we measure the averaged decoding latency with batch size set to 1 on the WMT14 En→De test set, using a NVIDIA GeForce RTX 3090 GPU.

[1] https://github.com/moses-smt/mosesdecoder.
[2] https://github.com/stanfordnlp/stanza.
[3] https://github.com/OpenNMT/OpenNMT-py.

4.3 Result

SDPSAT and a series of baselines are thoroughly compared in Table 1. The results lead to the following conclusions:

(i) SDPSAT attains state-of-the-art performance within Semi-NAT model group. SDPSAT outperforms the SAT baselines (GALN-SAT and GNLA-SAT) with better BLEU points and comparable speedup on all benchmark datasets. Also, SDPSAT achieves about 0.44/0.73 points improvement over its strong competitor RecoverSAT ($K = 2/5$) on average.

(ii) SDPSAT gains comparable performance with iterative NAT and fully NAT competitors. Compared with iterative NAT, SDPSAT (AT) achieves better results than CMLM and DisCo, while maintaining a greater speedup. As for fully NAT, except for REDER reaches a slightly higher BLEU on WMT16 Ro→En, SDPSAT achieves competitive performance with the remaining models.

Table 1. Performance comparison (BLEU scores and speedup rates) between SDPSAT and baselines. L denotes the deepest dividing layer, and M denotes the maximum segment size. K represents the segment number for SAT. G is the group size of GALN-SAT. NPD represents Noisy Parallel Decoding. n is the sample size of NPD. *iter* is an abbreviation for iteration. c is the maximum chunk size. * denotes the results under our implementation. The Mixed Training strategy is implemented for all of our SDPSAT models after training for 20 epochs for better convergence. "–" represents the data unreported.

Models		Speed	WMT14		WMT16	
			En→De	De→En	En→Ro	Ro→En
AT	Transformer	1.0×	27.30	–	–	–
	Transformer*	1.0×	27.48	31.88	33.69	33.98
Iterative NAT	CMLM (*iter* = 10) [3]	1.5×	27.03	30.53	33.08	33.31
	DisCo (*iter* = 10) [6]	1.5×	27.06	30.89	32.92	33.12
	LevT (*iter* = Adv.) [5]	4.0×	27.27	–	–	33.26
Fully NAT	NAT-FT+NPD ($n = 100$) [4]	2.4×	19.17	23.20	29.79	31.44
	SynSt ($c = 6$) [1]	4.86×	20.74	25.50	–	–
	FlowSeq+NPD ($n = 30$) [10]	1.1×	25.31	30.68	25.31	30.68
	CR-LaNMT [27]	11.0×	26.23	31.23	32.50	32.14
	ReorderNAT (AT) [16]	5.96×	26.49	31.13	31.70	31.99
	GLAT+NPD ($n = 7$) [14]	7.9×	26.55	31.02	32.87	33.51
	DCRF-NAT (rescoring 19) [19]	4.39×	26.80	30.04	–	–
	duplex REDER [26]	5.5×	27.36	31.10	33.60	**34.03**
Semi NAT	GALN-SAT ($G = 2$) [21]	2.07×	26.09	–	–	–
	GNLA-SAT ($K = 5$)*	4.49×	22.20	26.11	30.15	29.96
	GNLA-SAT ($K = 2$)*	1.97×	25.97	29.94	32.62	32.70
	RecoverSAT ($K = 5$) [15]	3.16×	26.91	31.22	32.81	32.80
	RecoverSAT ($K = 2$)	2.02×	27.11	31.67	32.92	33.19
	SDPSAT(NAT)($L = 2$, $M = 10$)	3.65×	25.30	30.51	32.84	32.64
	SDPSAT(NAT)($L = 1$)	2.94×	25.79	30.66	33.10	32.85
	SDPSAT(AT)($L = 2$, $M = 10$)	2.63×	26.88	31.12	33.39	33.32
	SDPSAT(AT)($L = 1$)	2.30×	**27.44**	**31.71**	**33.64**	33.85

In addition, SDPSAT enjoys a $2\times \sim 3\times$ speed advantage over the AT baseline while maintains close translation quality.

(iii) The last group of Table 1 shows that SDPSAT (AT) gets a better translation quality than SDPSAT (NAT) with a slight speedup drop, indicating that SDPSAT (AT) better captures the syntactic information than SDPSAT (NAT). What' s more, the inference of models with a larger deepest dividing layer could be better expedited at the expense of slightly degraded performance.

We perform experiments on WMT17 En↔Zh datasets with a huge linguistic gap, to further verify the effectiveness of our SDPSAT. Table 2 shows that SDPSAT achieves high BLEU scores in both directions, indicating that the syntactic decoder could provide syntactic structure hints for the target language on the basis of the source language despite the huge linguistic disparities.

Table 2. The performance comparison between SDPSAT and baselines on WMT17 En↔Zh.

Model	WMT17	
	Zh→En	En→Zh
AT Transformer*	24.30	35.44
CMLM($iter$ = 10)	23.21	33.19
LevT($iter$ = Adv.)	23.30	33.90
DisCo($iter$ = 10)	23.68	34.51
SDPSAT(AT)(L = 2,M = 10)	23.78	33.60
SDPSAT(AT)(L = 1)	24.00	34.53

5 Analysis and Discussion

The Effect of Mixed Training. To explore the effect of the rate controlling parameter γ in Mixed Training, we conducted experiments for SDPSAT(AT) ($L = 1$) on four validation sets. Table 3 shows that the parameter γ impacts the model's performance significantly. Mixed Training under appropriate parameter γ improves translation quality (*i.e.*, $\gamma = 10$), indicating that the strategy could alleviate *exposure bias*. However, too large γ (*i.e.*, $\gamma = 100$) may hurt the translation quality instead. With a larger γ, the model is more likely to be supervised incorrectly when it has not converged yet, which is detrimental to the training.

Table 3. The effect of γ on Mixed Training

γ	WMT14		WMT16	
	En→De	De→En	En→Ro	Ro→En
0	26.31	31.40	32.97	33.68
10	26.53	31.58	33.93	34.69
100	26.05	30.44	32.71	33.56

The Effect of Sentence Length. We group the WMT14 En→De validation set according to source sentence lengths and calculate their BLEU, respectively. Figure 3 shows that SDPSAT (AT) outperforms GNLA-SAT for most of the lengths. Meanwhile, when the sentence length is greater than 20 and less than 50, the translation performance of SDPSAT (AT) is very close to the AT model, which proves the effectiveness of SDPSAT.

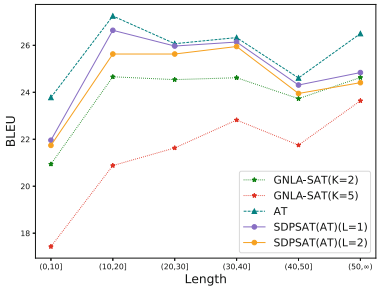

Fig. 3. Performance on various length groups.

The Effect of Repetition. Table 4 demonstrates that our GNLA-SAT which adopts equal-length segmentation, suffers from highly severe multi-modality problem, while SDPSAT reduces word repetition significantly. This finding indicates that the syntactic structure could make the meaning of each segment clearer, leading to the improvement of the translation quality. In addition, SDPSAT (AT) reaches a lower repetition rate than the strong competitor RecoverSAT.

Table 4. The performance (BLEU) and repeated rates on two benchmark validation sets.

Model	WMT14			
	En→De		De→En	
	BLEU	Reps	BLEU	Reps
SDPSAT (AT)	26.53	0.60%	31.58	0.30%
SDPSAT (NAT)	25.58	1.53%	30.54	1.43%
GNLA-SAT ($K = 2$)*	25.04	2.76%	29.93	2.97%
GNLA-SAT ($K = 5$)*	22.05	20.80%	26.53	19.07%
RecoverSAT ($K = 2$)*	26.17	1.43%	30.73	1.00%
RecoverSAT ($K = 5$)*	24.33	2.10%	28.65	1.77%

Case Study. Table 5 shows a translation comparison between SDPSAT with the SAT baselines. We find that (i) in GNLA-SAT, where the generated sentences are consisted of equal-length segments, the multi-modality problem is severe and the semantically consecutive words are often separated. In contrast, SDPSAT which relies on the syntax structure as initialization and temination criterion, generates more semantically consistent segments and more fluent translation. (ii) in RecoverSAT, due to implicit segmentation, the length of the generated segment varies greatly, which harms the decoding efficiency. In RecoverSAT ($K = 2$), the model generates an empty segment (*i.e.*, `[BOS] [EOS]`) and it even degrades to AT model. What' s more, RecoverSAT requires additional operation of removing duplicate segments (*e.g.*, enough). In comparison, the decoding pattern of SDPSAT is clearer and more straightforward.

Table 5. Translation comparison of SAT baselines and SDPSAT(AT)(L=1; $L = 2$, $M = 10$). [BOS] and syntactic label (e.g., [obl]) denotes the segment beginning. The [EOS] and [DEL] denotes the operation of keeping or deleting the segment for RecoverSAT. We use the double underscores (e.g., is) for repeated tokens, dashed underline (e.g., _ _) for missing words, and wave underline (e.g., targets) for semantic errors.

Source		Wer sich weniger als fünf Minuten ge@@ dul@@ det, wartet unter Umständen nicht lange genug, warnt Bec@@ ker und verweist auf einen Beschluss des Ober@@ land@@ es@@ geri@@ chts Ham@@ m
Reference		Those who tolerate less than five minutes may not wait long enough, warns Becker, referring to a decision of the Hamm Higher Regional Court
+Syntactic label		[nsubj:1] Those who tolerate less than five minutes [aux:1] may [advmod:1] not wait [advmod:1] long [advmod:1] enough [parataxis:1], warns Becker [conj:1] [cc:2] and refers [obl:2] to a decision of the Hamm Higher Regional Court [punct:1].
GNLA-SAT	K = 2	[BOS] Anyone who is tolerated less than five minutes may not wait long enough [BOS], warns Becker and refers to a decision by the Supreme _ _ Court _ _ Hamm
	K = 5	[BOS] Anyone who condonates less than five [BOS] five minutes may not wait for [BOS] wait for a long enough, [BOS] _ _ Becker and points to a Hamm [BOS] decision by the Supreme _ _ Cour
Recover-SAT	K = 2	[BOS] Anyone who tolerates less than five minutes may not wait long enough, warns Becker and refers to a decision of the Hamm Supreme _ _ Court. [EOS] [BOS] [EOS]
	K = 5	[BOS] Anyone who tolerates [EOS] [BOS] less than five minutes may not wait long [EOS] [BOS] enough, warns Becker and refers to a decision by the Supreme _ _ Court _ _ Hamm. [EOS] [BOS] enough [DEL] [BOS] [EOS]
Ours	L = 1	[nsubj] Anyone who tolerates less than five minutes [aux] may [advmod] not wait [advmod] long enough [conj], warns Becker [conj] and refers to a decision of the Higher Regional Court of Hamm [punct]
	L = 2	[nsubj:1] Those who tolerate less than five minutes [aux:1] may [advmod:1] not wait [advmod:1] long enough [parataxis:1], warns Becker [conj:1] [cc:2] and refer [obl:2] to a decision by the Hamm Supreme _ _ Court [punct:1]

6 Conclusion

We develop a layer-and-length-based syntactic labeling approach and present a novel two-stage SAT framework called SDPSAT, which enables a more flexible termination and a better initialization for SAT decoding. Besides, we present a Mixed Training strategy to diminish the exposure bias. Experimental results suggest that SDPSAT excels within the Semi-NAT group and achieves comparable translation performance with those strong NAT or AT competitors, while significantly alleviating the multi-modality problem.

Acknowledgements. This research work has been funded by Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020), and Shanghai Science and Technology Plan (Grant No. 22511104400).

References

1. Akoury, N., Krishna, K., Iyyer, M.: Syntactically supervised transformers for faster neural machine translation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 1269–1281 (2019)
2. Chen, K., Wang, R., Utiyama, M., Sumita, E., Zhao, T.: Syntax-directed attention for neural machine translation. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4792–4799 (2018)
3. Ghazvininejad, M., Levy, O., Liu, Y., Zettlemoyer, L.: Mask-predict: parallel decoding of conditional masked language models. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 6112–6121 (2019)
4. Gu, J., Bradbury, J., Xiong, C., Li, V.O.K., Socher, R.: Non-autoregressive neural machine translation. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3 2018, Conference Track Proceedings (2018)
5. Gu, J., Wang, C., Junbo, J.Z.: Levenshtein transformer. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 11181–11191 (2019)
6. Kasai, J., Cross, J., Ghazvininejad, M., Gu, J.: Non-autoregressive machine translation with disentangled context transformer. In: International Conference on Machine Learning, pp. 5144–5155 (2020)
7. Le, A.N., Martinez, A., Yoshimoto, A., Matsumoto, Y.: Improving sequence to sequence neural machine translation by utilizing syntactic dependency information. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 21–29 (2017)
8. Li, Y., Cui, L., Yin, Y., Zhang, Y.: Multi-granularity optimization for non-autoregressive translation. arXiv preprint arXiv:2210.11017 (2022)
9. Liu, Y., Wan, Y., Zhang, J., Zhao, W., Philip, S.Y.: Enriching non-autoregressive transformer with syntactic and semantic structures for neural machine translation. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 1235–1244 (2021)
10. Ma, X., Zhou, C., Li, X., Neubig, G., Hovy, E.: FlowSeq: non-autoregressive conditional sequence generation with generative flow. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4282–4292 (2019)
11. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318 (2002)
12. Post, M.: A call for clarity in reporting bleu scores. In: Proceedings of the Third Conference on Machine Translation: Research Papers, pp. 186–191 (2018)
13. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: a python natural language processing toolkit for many human languages. In: Proceedings of the

58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 101–108 (2020)
14. Qian, L., et al.: Glancing transformer for non-autoregressive neural machine translation. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1993–2003 (2021)
15. Ran, Q., Lin, Y., Li, P., Zhou, J.: Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3059–3069 (2020)
16. Ran, Q., Lin, Y., Li, P., Zhou, J.: Guiding non-autoregressive neural machine translation decoding with reordering information. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 13727–13735 (2021)
17. Ranzato, M., Chopra, S., Auli, M., Zaremba, W.: Sequence level training with recurrent neural networks. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016, Conference Track Proceedings (2016)
18. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715–1725 (2016)
19. Sun, Z., Li, Z., Wang, H., He, D., Lin, Z., Deng, Z.: Fast structured decoding for sequence models. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, pp. 3011–3020 (2019)
20. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
21. Wang, C., Zhang, J., Chen, H.: Semi-autoregressive neural machine translation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 479–488 (2018)
22. Xiao, Y., et al.: A survey on non-autoregressive generation for neural machine translation and beyond. arXiv preprint arXiv:2204.09269 (2022)
23. Zhang, K., et al.: A study of syntactic multi-modality in non-autoregressive machine translation. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1747–1757 (2022)
24. Zhang, W., Feng, Y., Meng, F., You, D., Liu, Q.: Bridging the gap between training and inference for neural machine translation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4334–4343 (2019)
25. Zhang, Z., Wu, Y., Zhou, J., Duan, S., Zhao, H., Wang, R.: SG-Net: syntax guided transformer for language representation. IEEE Trans. Pattern Anal. Mach. Intell. **44**(06), 3285–3299 (2022)
26. Zheng, Z., Zhou, H., Huang, S., Chen, J., Xu, J., Li, L.: Duplex sequence-to-sequence learning for reversible machine translation. Adv. Neural. Inf. Process. Syst. **34**, 21070–21084 (2021)
27. Zhu, M., Wang, J., Yan, C.: Non-autoregressive neural machine translation with consistency regularization optimized variational framework. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 607–617 (2022)