A Survey on Non-Autoregressive Generation for Neural Machine Translation and Beyond

Yisheng Xiao*, Lijun Wu*, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, *Senior Member, IEEE* and Tie-Yan Liu, *Fellow, IEEE*

Abstract—Non-autoregressive (NAR) generation, which is first proposed in neural machine translation (NMT) to speed up inference, has attracted much attention in both machine learning and natural language processing communities. While NAR generation can significantly accelerate inference speed for machine translation, the speedup comes at the cost of sacrificed translation accuracy compared to its counterpart, auto-regressive (AR) generation. In recent years, many new models and algorithms have been designed/proposed to bridge the accuracy gap between NAR generation and AR generation. In this paper, we conduct a systematic survey with comparisons and discussions of various non-autoregressive translation (NAT) models from different aspects. Specifically, we categorize the efforts of NAT into several groups, including data manipulation, modeling methods, training criterion, decoding algorithms, and the benefit from pre-trained models. Furthermore, we briefly review other applications of NAR models beyond machine translation, such as dialogue generation, text summarization, grammar error correction, semantic parsing, speech synthesis, and automatic speech recognition. In addition, we also discuss potential directions for future exploration, including releasing the dependency of KD, dynamic length prediction, pre-training for NAR, and wider applications, etc. We hope this survey can help researchers capture the latest progress in NAR generation, inspire the design of advanced NAR models and algorithms, and enable industry practitioners to choose appropriate solutions for their applications. The web page of this survey is at https://github.com/LitterBrother-Xiao/Overview-of-Non-autoregressive-Applications.

Index Terms—Non-autoregressive, Neural Machine Translation, Transformer, Sequence Generation, Natural Language Processing

1 Introduction

Machine translation [1] is one of the most critical and challenging tasks in natural language processing (NLP), which aims to translate natural language sentences from the source language to the target language. Recently, with the breakthrough of deep learning [2], Neural Machine Translation (NMT) [3], [4], [5], [6], [7], [8], which takes the different neural networks as backbone models, e.g., RNN [3], [9] and CNN [10], [11], has achieved outstanding performances, especially for the self-attention [12] based Transformer [13] models [14], [15]. NMT usually adopts the auto-regressive generation (AR) method for translation (AT), which means the target tokens are one-by-one generated in a sequential manner. Therefore, AT is quite time-consuming when generating target sentences, especially for long sentences. To alleviate this problem and accelerate decoding, non-autoregressive generation (NAR) for machine translation (NAT) is first proposed in [16], which can translate/generate all the target tokens in parallel. Therefore, the inference speed is hugely increased, and much attention to NAT/NAR methods has been attracted with impressive progress [17], [18], [19], [20], [21], [22], [23]. However, the translation accuracy is damaged and sacrificed as a result of parallel decoding. Compared with AT, the tokens are generated without internal dependency for NAT models, unlike the AT models where the t-th token has previous $t-1$ contextual token information to help its generation. Hence, the NAT models seriously suffer from lacking target side information to make predictions (e.g., decoding length) and correctly generate target translations. In summary, we attribute the main challenge of NAT models to the *'failure of capturing the target side dependency.'*

To mitigate the above-mentioned challenge, significant efforts have been paid in the past few years from different aspects, e.g., data manipulation [20], [24], modeling methods [18], [25], decoding strategies [26], [27], to better capture the dependency on target side information. Although impressive progress has been achieved and the translation accuracy is greatly improved for NAT models, the translation quality still falls behind their AT counterparts. To continue narrowing the performance gap and facilitating the development of NAT in the future, a solid review of current NAT research is necessary. Therefore, we make the first comprehensive survey of existing non-autoregressive technologies for NMT in this paper. Our review summarizes the core challenge of NAT research and presents various advanced approaches to solve the challenge. Specifically, we introduce the approaches from the following aspects:

- **Data Manipulation.** As a data-driven task, the scale and quality of training data are crucial for NMT tasks. Due to the lack of target dependency for NAT models, lots of methods are proposed to reduce the complexity of the training data to provide an easier training task for them.
- **Modeling.** Various advanced model architectures are proposed to better capture the target dependency, including iteration-based methods that can provide partially

- *Yisheng Xiao, Juntao Li, and Min Zhang are with Soochow University, Suzhou, China. E-mail: ysxiaoo@stu.suda.edu.cn; ljt@suda.edu.cn; minzhang@suda.edu.cn.*
- *Lijun Wu, Junliang Guo, Tao Qin, and Tie-Yan Liu are with Microsoft Research, Beijing 100080, China. E-mail: lijuwu@microsoft.com; junliangguo@microsoft.com; taoqin@microsoft.com; tyliu@microsoft.com.*
- *Yisheng Xiao and Lijun Wu contribute equally to this paper.*

Manuscript received April 20, 2022

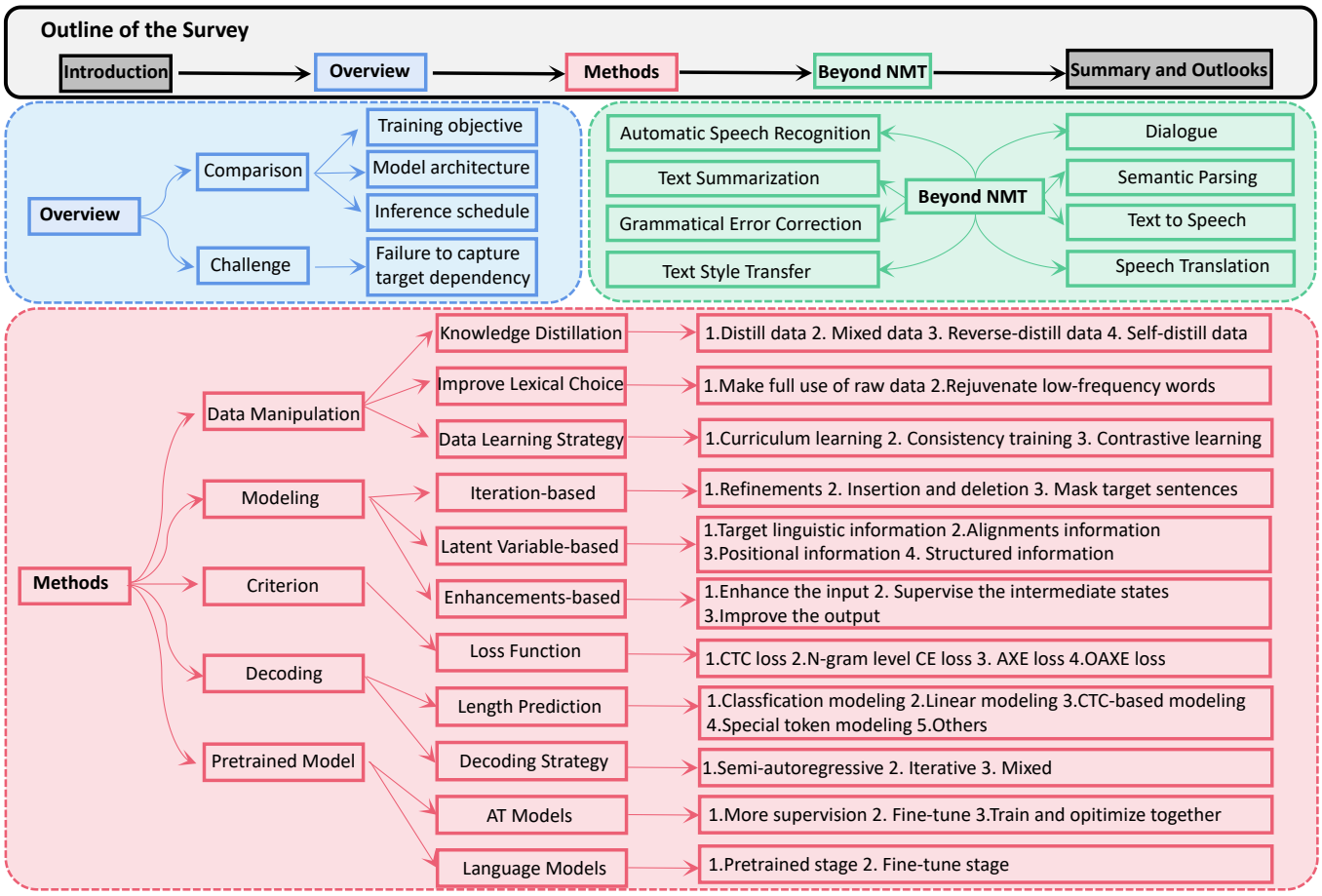


Fig. 1. Outline of the survey. We first review the developments of neural machine translation and non-autoregressive related methods. Then we present an overview of recent AT and NAT models, including a comparison of three different aspects (i.e., training objective, model architecture, and inference schedule). Besides, we also summarize the main challenge that recent NAT models encounter compared with AT models. To solve these problems, we introduce several widely used methods to help improve the ability of NAT models at different levels, including data manipulation, modeling, criterion, decoding, and benefiting from pre-trained models. Then we present a summary of the above methods for the machine translation task. We also extend this survey to other extensive applications, such as automatic speech recognition, text summarization, grammatical error correction, text style transfer, dialogue, semantic parsing, text to speech and speech translation. Finally, some open problems and future outlooks for the non-autoregressive technique are discussed. Best viewed this outline in color.

observed target information, latent variable-based methods that introduce latent variables to learn the target-side dependency, and enhancements-based methods that directly provide stronger target-side information to the input/output/intermediate states of the decoder.

- **Criterion.** Some works point out that the traditional cross-entropy loss is not optimal for NAT models and propose better criteria to improve the training of NAT models, including Connectionist Temporal Classification (CTC) based, N-gram based, and order-based loss functions.
- **Decoding.** Decoding algorithm is another decisive factor in NMT models. Upon NAT models, different tricks are proposed to improve the decoding process and provide better translation results.
- **Pre-Trained Model.** Finally, given the strong representation capacity of pre-trained models, it is appealing to utilize them to improve the performance of NAT models. Therefore, lots of methods have been proposed to leverage the information from pre-trained AT models or large-scale language models to help the training of NAT models.

Besides summarizing the improving works for NAT, we also review other applications of NAR methods beyond NMT, such as dialogue generation, text summarization, semantic parsing, grammar error correction, text to speech, speech translation, and automatic speech recognition. We further point out the recent trends and possible promising directions for future development, such as releasing the dependency of knowledge distillation, designing dynamic length prediction mechanisms, and pre-training for NAR models. We hope this survey paper can provide researchers and engineers with valuable insights to attract more people to either promote the development of NAT/NAR techniques or bring NAT/NAR methods into other fields, such as NAR text generation with large-scale pre-trained language models. Besides, the up-to-the-minute solutions for each NAT problem and thorough analysis of model performance and computational cost are also expected to assist considerable industry practitioners.

The organization of this survey paper is as follows. To begin with, we make a comparison between AT and NAT models from different views, such as their training objectives, model architectures, and inference schedules. Then we analyze the main challenge that the current NAT models

encounter in Section 2. We attribute the low quality of NAT models to the failure to well capture the target dependency, and from Section 3 to Section 7, we summarize the efforts paid for improvement from different aspects, including data manipulations (Section 3), modeling methods (Section 4), training criterion (Section 5), decoding ways (Section 6), and the benefit from pre-trained models (Section 7). In addition, we also give a short summary of current NAT models in Section 8. Next, we investigate the extension of NAR generation approaches to other various applications beyond NMT in Section 9. At last, we conclude this paper and discuss our future outlooks in Section 10. A detailed version of our survey outline is also shown in Figure 1.

2 Overview of AT and NAT Models

In this section, we first give an overall introduction of AT and NAT models to better understand the necessary background. We briefly compare them from several different aspects, including the training objective, model architecture, and inference strategy. Besides, we also analyze the main challenge that NAT models encounter compared with AT.

2.1 Comparison

Both AT and NAT models aim to make a correct sentence translation from a source language to a target language. Due to their unique characteristics, their differences are apparent in training, modeling, and inference. Before a detailed comparison, we first introduce the necessary notations.

Given a dataset $D = \{(X, Y)_i\}_{i=1}^{N}$, where $(X, Y)_i$ refers to a paired sentence data, and N is the size of the dataset. X is the sentence to be translated from source language $\mathcal{X}$ and Y is the ground-truth sentence from target language $\mathcal{Y}$. The goal of NMT models is to learn a mapping function $f(\cdot)$ from the source sentence to the target sentence $f : X \to Y$ to estimate the unknown conditional distribution $P(Y|X;\theta)$, where θ denotes the parameter set of a network model. We now compare the details of AT and NAT models as below.

Training Objective. (1) For paired sentences (X, Y), where $X = \{x_1, x_2, ..., x_{T_X}\}$ and $Y = \{y_1, y_2, ..., y_{T_Y}\}$, the training objective $\mathcal{L}_{\text{AT}}$ of an auto-regressive NMT (AT) model is to maximize the following likelihood:

$$\mathcal{L}_{\text{AT}} = \sum_{t=1}^{T_Y} \log P(y_t|y_{<t}, X;\theta),$$

where y_t is the token to be translated at current time step t and $y_{<t}$ are the tokens predicted in previous $t-1$ decoding steps. From the above equation, we can clearly see that the training of AT models adopts the auto-regressive factorization in a left-to-right manner. Note that during training, the ground-truth target tokens are leveraged with the teacher forcing method [13], [28]. In this way, the translation quality is guaranteed with the help of contextual dependencies.

(2) In contrast, the non-autoregressive NMT (NAT) models [16] use the conditional independent factorization for prediction, and the objective is to maximize the likelihood:

$$\mathcal{L}_{\text{NAT}} = \sum_{t=1}^{T} \log P(y_t|X;\theta),$$

notice that T is the length of the target sentence. During training, $T = T_Y$ is the length of the ground-truth target sentence, while in inference, $T = P_L(X)$ which is usually predicted by a length prediction module P_L. Compared with AT models, it is obvious that the conditional tokens $y_{<t}$ are removed for NAT models. Hence, we can do parallel translations without auto-regressive dependencies, and the inference speed is greatly improved.

(3) Besides the AT and NAT models, researchers aim to find an intermediate state between current AT and NAT, which can also serve as a universal formulation of both models to achieve a balance between decoding speed and translation quality. For example, Wang *et al.* [17] propose a semi-autoregressive NMT (SAT) model, which keeps the auto-regressive property in global but relieves it in local. Shortly speaking, SAT models can produce multiple target tokens in parallel at each decoding step (local non-autoregressive) and dependently generate tokens for the next step (global auto-regressive). Mathematically, SAT models aim to maximize the following likelihood:

$$\mathcal{L}_{\text{SAT}} = \sum_{t=1}^{[(T-1)/k]+1} \log P(G_t|G_{<t}, X;\theta),$$

where k denotes the number of the tokes that the SAT models parallelly generate at one time step. G_t is a group of k target tokens at t-th step. $G_{<t}$ is the $t-1$ groups of target tokens generated in the previous $t-1$ decoding steps. Note that if $k = 1$, it equals an AT model, and if $k = T$, it generalizes to a NAT model.

(4) As a comparison, iterative-based NAT models share a similar spirit of mixed auto-regressive and non-autoregressive translation, but on the sentence level with a refinement approach. That is, iterative-based NAT models keep the non-autoregressive property in every iteration step and refine the translation results during different iteration steps [18], [29]. The training goal of the iterative-based NAT models is to maximize:

$$\mathcal{L}_{\text{Iter}} = \sum_{y_t \in Y_{tgt}} \log P(y_t|\hat{Y}, X;\theta),$$

where $\hat{Y}$ indicates the translation result of the last iteration, and Y_{tgt} is the target of this iteration.

In the first iteration, only X is fed into the model, which is the same as NAT models. After that, each iteration takes the translation generated from the last iteration as context for refinement to decode the translation. Generally speaking, NAT models with iterative refinements are viewed as iterative-based NAT models, while models with only one decoding step are viewed as fully NAT models.

Model Architecture. As for model architecture, both AT and NAT models take the encoder and decoder framework for translation. The encoder and decoder can be different neural networks, such as RNN [9], CNN [11], and Transformer [13]. Due to the superior performance of the Transformer network, we focus on the Transformer model for discussion in this survey. The encoder is used to encode the source sentences, while the decoder is utilized for decoding the target sentence. Compared to AT and NAT models, they adopt the same encoder architecture, and the differences are reflected in the decoders to match the specific training objective. (1)

Specifically, AT models need to prevent earlier decoding steps from peeking at information from later steps. Therefore, the constraint of an auto-regressive factorization of the output distribution is required, and they adopt the strict causal mask by applying a lower triangular matrix in the self-attention module of the conventional Transformer decoder [13]. (2) However, for NAT models, including the iterative-based NAT models, this constraint is no longer necessary, so they adopt the unmasked self-attention over all target tokens [16]. (3) As for SAT models, they adopt a coarse-grained lower triangular matrix as the causal mask, which means that they allow k tokens to peep later information in the same group while keeping the constraint between different groups.

Inference Schedule. When going to the inference stage, the differences are as follows. (1) The AT models predict the target tokens in a one-by-one manner, and the tokens predicted previously are fed back into the decoder to generate the next token. (2) While SAT models predict a group of target tokens at one time, the previously generated groups of tokens are fed into the decoder to generate the next group of tokens, which is the same as the AT models. (3) For iterative-based NAT models, it needs k iterations for inference. The translated results of the previous iteration will be fed into the decoder again for refinements. (4) As for fully NAT models, they generate all predicted target tokens at only one step, which greatly speeds up inference. It is worth noting that AT and SAT models suffer from the gap between training and inference [30], [31], [32]. That is, they utilize ground-truth target tokens during training, while the models can only take previously generated target tokens for inference. This indeed leads to inconsistency between training and inference and hence hurts the their performance. In contrast, fully NAT models are free from this trouble, but for iterative-based NAT models, prediction in the previous iteration is adopted for refinements, and this mismatched problem may be more serious. More details about this will be discussed in Section 6.

2.2 The Main Challenge of NAT Models

When achieving parallel decoding, a critical issue of NAT models is that they have no tokens with target information fed into the decoder [16] during training and inference. They can only rely on the source side information, which heavily increases the difficulty for NAT models. Previously, when Gu *et al.* [16] first propose their NAT model, they notice that using nothing or only position embeddings in the first decoder layer results in poor translation performance. To alleviate this problem, they propose an initial module by copying the source tokens as the initialization for the decoder input. However, the source and target sentences are indeed different from distinct languages. This way does not help the decoder since no target information is given.

As a result, missing the target information leads the NAT models *to fail to capture the target dependency of target tokens*, and we attribute the main challenge of low quality for NAT models to this defect. To better understand and further release this problem, we now give specific analysis with examples and also briefly show improvement methods in the following contents.

Understanding the Problem. Since no target information is fed into the decoder, NAT models remove the word dependency of the target sentence completely and generate target tokens entirely depending on the source sentence. Hence, terrible situations can happen to harm the translation quality. (1) First, the conditional independence assumption prevents a model from properly capturing the highly multi-modal distribution of target translations, which is called multi-modality problem [16]. Almost all the NAT models suffer from this trouble. Due to the strong assumption that each target token is predicted independently, if there are several different target sentences that can be viewed as reasonable translations, NAT models are possible to select fragments of each sentence and combine them as a candidate translation. Take an example, when translating `thank you` into German, `Vielen Dank` and `Danke` are both reasonable translations. However, NAT models may generate `Danke Dank`, which is truly unreasonable but should be impossible in AT models. (2) Over-translation and under-translation [22] are also common translation errors. The issue of over-translation refers to the same word token being successively generated multiple times, leading the same token from different reasonable translations to appear at different positions in the final translation. The under-translation indicates that several necessary tokens in the source sentence are neglected, leading to several tokens missing in the translation results. Take an example, when translating German sentence `es gibt heute viele Farmer mit diesem Ansatz` into English sentence, a reasonable translation can be `there are lots of farmers doing this today`. However, NAT models may miss the word `of` (under-translation) or generate the word `of` twice (over-translation), leading the results to be `there are lots farmers doing this today` or `there are lots of of farmers doing this today`. This seriously harms the translation quality. Instead, if target dependency is given as AT models, the problem of repetitive tokens and missing tokens can be avoided.

2.3 Overview of Improving Methods

As we discussed that NAT models are hard to model the target side dependency, various methods have been proposed to alleviate this problem by reducing the dependency of target tokens at different levels, which hence improves the ability of NAT models. To have a clear overview of these methods, we show the general framework and the data flow of various NAT models in Figure 2, which contain different components such as the data preparation, NAT encoder, and NAT decoder. The other improvement methods we will introduce later can be summarized to focus on these different components. Specifically, these methods include: (1) data manipulation, which focuses on the improvements of training data corpus and data learning strategies, (2) improvements on the modeling level, where we first summarize two popular and widely used training frameworks (iteration-based methods and latent variable-based methods) along with various specific implementations of them. Besides, various other enhancements-based methods are introduced for NAT models, (3) improvements on the training criterion, where better criteria compared with traditional cross-entropy loss are proposed to meet the unique characteristics of NAT models, (4) improvements on the decoding level, where

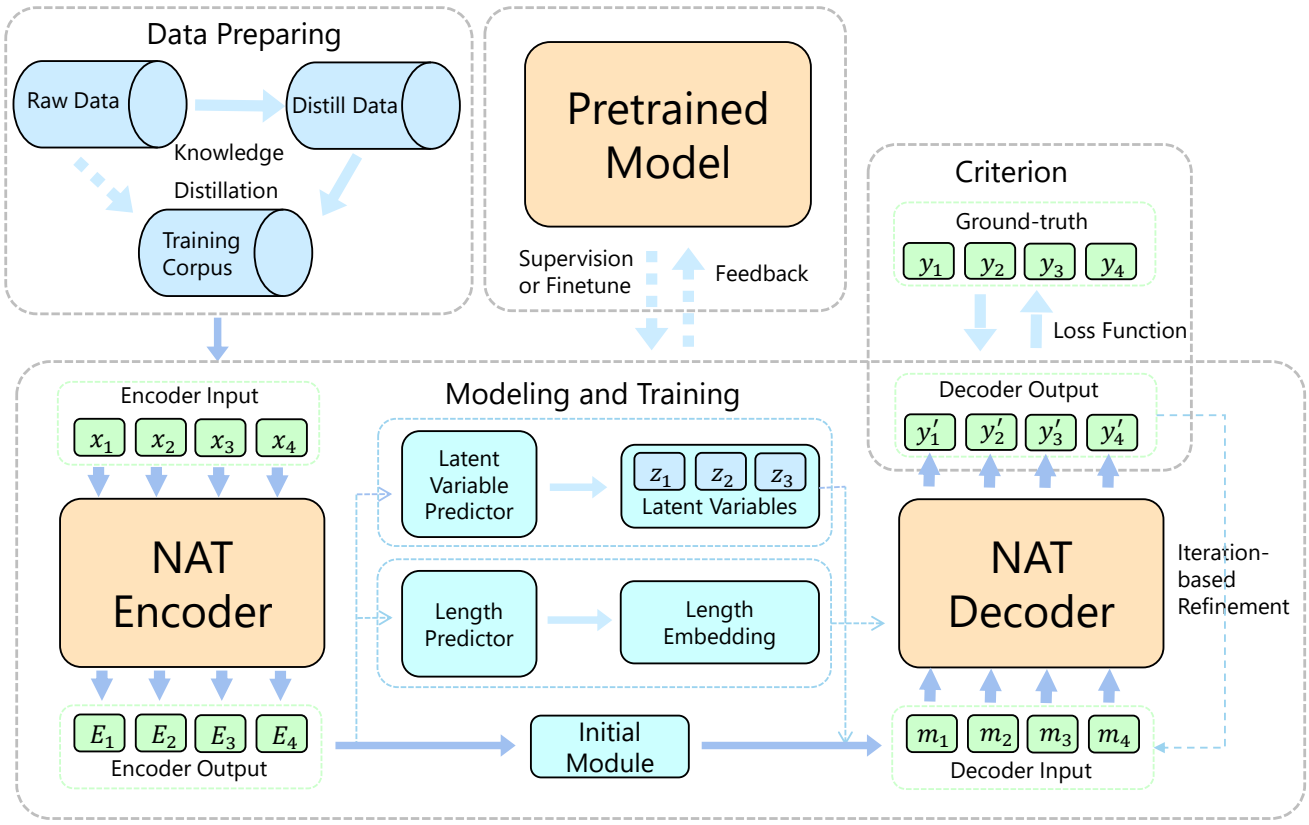


Fig. 2. The overall framework of various NAT models. The dashed arrow denotes this part is not applied in all NAT models. Different knowledge distillation methods will be applied in Data Preparing part. Length Predictor is applied in most NAT models, and Latent Variable Predictor is applied in latent variable-based models. Initial Module is used to initialize the Decoder Input, such as soft-copy, source-copy, partial target tokens, etc.

we introduce the tremendous progress made on length prediction and decoding strategy, and (5) benefiting from pre-trained models, i.e., guiding NAT models to benefit from their AT counterparts and other large scale pre-trained language models such as BERT [33].

In Table 1, we give a summary and overview of different NAT models based on the above improvement category. In each category, we also present the specific sub-topics to better classify the category, along with the representative published works. For example, the decoding strategies can be divided to semi-autoregressive decoding (i.e., Semi-NAT [17]), iterative decoding (i.e., Easy-first [34]), and also mixed decoding (Unify [35]). For each work, we summarize a short description and list its published place (e.g., ACL, EMNLP), the decoding speed, and the performance on the mostly evaluated dataset WMT14 English→German (EN→DE) for a quick understanding.

Before introducing these methods, in Figure 3, the most important and popular works along the NAT development are shown in the timeline. The NAT is first proposed in November 2017, and the inference speed is hugely improved, but the accuracy is far behind the AT model. After its born, Semi-NAT [17] is proposed to serve as the bridge between AT and NAT models with better translation performance. Other representative works are then introduced, including iterative-based methods: CMLM [18] and Imputer [36], fully NAT models: GLAT [37] and Fully NAT [38]. These models mainly conduct improvements to the model structure. Besides, improvements based on training criteria are also introduced in OAXE NAT [39]. Recently, CeMAT [40] is proposed to explore the potential of pre-training a non-autoregressive model and then fine-tuning on the translation task. With the rapid growth of NAT models, their performance gap with AT models is narrowing, and the tendency of developing NAT models in real-world systems is increasing. We will elaborate these methods in the following sections.

3 Data Manipulations

Neural machine translation is a data-driven task, and the performance of the NAT model heavily relies on the volume as well as the quality of the bilingual training data. Therefore, various data manipulation methods are proposed to help the model better capture the target side dependency. In this section, we will introduce these methods from two perspectives: (1) knowledge distillation which aims to reduce the complexity of the training corpus; (2) data learning strategies that help the model better learn and understand the training data. The introduced methods are listed in the "Data Manipulations" category of Table 1.

3.1 Knowledge Distillation

Initially, Knowledge Distillation (KD) [41] is proposed to train a weaker student model with soft prediction distributions generated by a stronger teacher model. Sequence-level knowledge distillation [42] extends it to the sentence level, where a pre-trained teacher model predicts sequences of tokens that are taken as the targets of the student model.

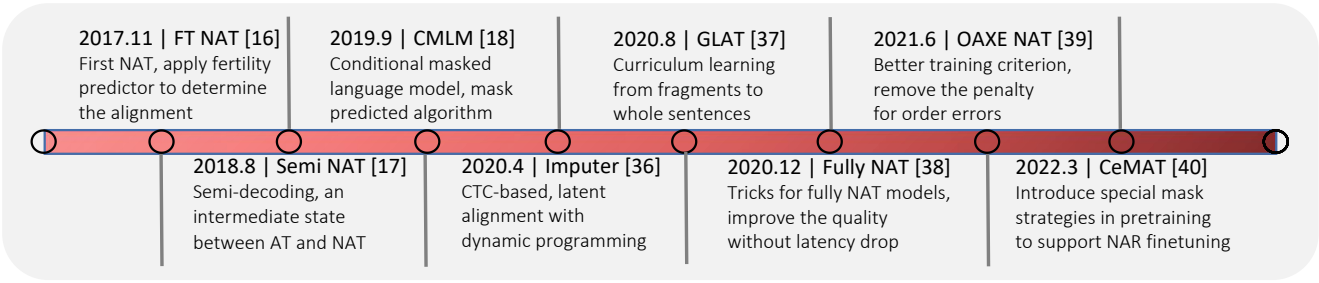


Fig. 3. Popular methods in the development of NAT models.

When applied to NAT models, a pre-trained AT model is utilized to generate distilled target sentences for all source sentences in the training set, with either greedy decoding or beam search. For example, given a pre-trained AT model θ_{AT} and the training set $D = \{(X, Y)_i\}_{i=1}^{N}$, the distilled target sentences Y' are generated as:

$$Y' \sim P(Y|X; \theta_{\text{AT}}),$$

where Y' is the decoding results of the pre-trained AT model with various decoding algorithms such as greedy decoding and beam search decoding. Then, we train the NAR models on this distilled training set $D' = \{(X, Y')_i\}_{i=1}^{N}$ with the traditional negative log-likelihood loss:

$$\mathcal{L}_{\text{KD}} = -\sum_{(x,y')\in\mathcal{D}'} \log P(y'|x; \theta_{\text{NAT}}),$$

where θ_{NAT} is the parameter set of the NAT model. KD is widely adopted as the distilled corpus is regarded as less noisy and more deterministic than the original one. To investigate the reason behind this, we review related works and give a detailed analysis from two aspects: (1) why is KD effective for NAT models? (2) does there exist drawbacks to current KD methods, and how to solve them?

Understanding Knowledge Distillation. Zhou *et al.* [43] propose two quantitive measures, including the complexity and faithfulness, to analyze the property of distilled data and its correlation with NAT performance. Specifically, they find that while KD generally simplifies the training data by reducing the complexity and increasing the faithfulness, a larger and better teacher model does not always lead to a better student model. Instead, the capacity of the teacher model should be aligned with the student NAT model to achieve the best performance. In addition, Ren *et al.* [44] design a model to measure the target token dependency over the data and find that KD can reduce the dependency when predicting target tokens, which is helpful for the training of NAT. Xu *et al.* [19] find that KD can also reduce the lexical diversity and word reordering degree, which helps the model better learn the alignment between source and target.

Problem and Improvements. Despite the effectiveness, there exist some problems when utilizing KD. Zhou *et al.* [43] find that the capacity of the NAT model should be correlated with the complexity of the distilled dataset. Therefore they propose several methods, including born-again network [45] and mixture-of-experts [46] to adjust the complexity of the dataset w.r.t the model capacity. In addition, after knowledge distillation, the density estimation of real data may be harmed [47] and the lexical choice may be mistaken [20]. Specifically, Ding *et al.* [20] suppose that the distill data mainly focuses on the performance of high-frequency words. They propose two evaluation metrics to measure the lexical translation accuracy and conclude that the accuracy of low-frequency words is seriously decreased when the NAT model is trained on distilled datasets. To deal with this problem, Ding *et al.* [48] make full use of raw, distill, and reverse-distill data to rejuvenate low-frequency words. Zhou *et al.* [49] add monolingual data for training of the teacher model to enrich the distill dataset. The effect and improvement of self-distillation are also explored [21].

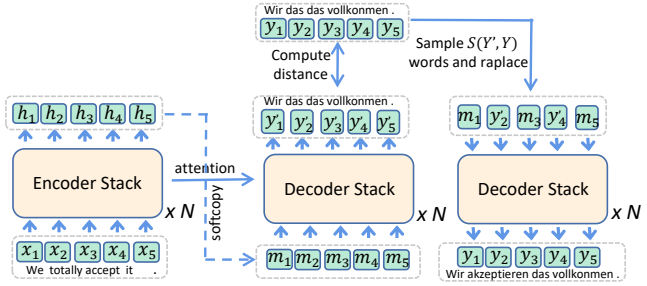


Fig. 4. An illustration of the model structure of Glat [37] in Section 3.2, which draws lessons from the idea of curriculum learning and aim to learn from fragments first and then from whole sentences gradually.

3.2 Data Learning Strategies

Aside from constructing informative training datasets, designing suitable learning strategies is another way to improve NAT models. We introduce various data learning strategies in this subsection. Curriculum learning [82] is a machine learning strategy inspired by human learning, which trains the model by feeding training instances in an order (e.g., from easy to hard) instead of randomly. Guo *et al.* [24] introduce the idea of curriculum learning into the training of NAT models by progressively switching the decoder input from AT to NAT to provide a smooth transformation between two training strategies. Liu *et al.* [79] extend this method by designing more fine-grained curriculums. Qian *et al.* [37] propose an adaptive glancing sampling strategy to guide the model to learn from fragments first and then from whole sentences gradually, shown in Figure 4. The ratio of fragments is correlated with the capacity of the model at the current training stage. Bao *et al.* [50] further extend

TABLE 1
A summary and overview of different NAT models discussed in this paper. The numbers of iteration, decoding speedup, and performance are all copied from their original paper. Specifically, "Performance" denotes the BLEU scores on the WMT 14 EN→DE dataset, and "Speedup" refers to the decoding speedup ratio compared with AT model. Note that "*" indicates training with sequence-level knowledge distillation from a big Transformer. The speedup may not be comparable due to their different hardware conditions, and we list them here just for reference. "#" denotes ACL Findings.

Category	Sub-category	Method	Description	Publication	Iteration	Speedup	Performance
Data Manipulations	Improving KD	MD [49]	Add monolingual data, enrich distillation corpus	ACL 2020	1	-	25.73
		RDP [20]	Raw data prior training, improve lexical choice	ICLR 2020	2.5	3.5x	27.80*
		LRF [48]	Add reverse-distill data,rejuvenate low-frequency word	ACL 2021	2.5	3.5x	28.20*
		SDMRT [21]	Self-distillation mixup, pre-rerank and fine-tune training	ARXIV 2021	10	-	27.72*
	Learning Strategies	Glat [37]	Glacing, learn from fragments to whole sentence	ACL 2021	1	15.3x	25.21
		latent-Glat [50]	Introduce glacing strategy to discrete latent variables	ACL 2022	1	11.3x	26.64
		PMG [51]	Multi-granularity, from words, phrases, sentences gradually	ACL# 2021	3.5	-	27.80*
		MvCR-NAT [52]	Consistency-based, masked token and model level consistency	ARXIV 2021	10	3.6x	27.39*
Modeling	Iteration-based methods	RefineNAT [29]	Denoising autoencoders, iterative refinement	EMNLP 2018	10	1.5x	21.61
		Insertion Transformer [53]	Insert tokens each iteration, like balanced binary trees	ICML 2019	$\approx log_2(N)$	-	27.41
		Levenshtein [54]	Insert and delete tokens during each iteration	NeurIPS 2019	Adaptive	4.0x	27.27
		CMLM [18]	Masked language model trained with uniform mask strategy	EMNLP 2019	10	1.7x	27.03*
		Disco [34]	More visible subsets to predict masked tokens	ICML 2019	Adaptive	3.5x	27.34*
		JM-NAT [55]	Jointly masked strategy, N-gram level masking in decoder	ACL 2019	10	5.7x	27.69*
		Imputer [36]	Combine conditional masking with CTC	EMNLP 2020	8	3.9x	28.20*
		Rewrite-NAT [26]	Reviewer and locator, locate the error and rewrite	EMNLP 2021	2.7	3.9x	27.83*
		CMLMC [56]	CMLM with reveal-position and correction function	ICLR 2022	10	-	28.37*
	Latent variable -based methods	FT-NAT [16]	Fertility predictor, determine the latent alignments	ICLR 2018	1	15.6x	17.69
		FlowSeq [57]	Generative flow, a powerful mathematical framework	EMNLP 2019	1	1.1x	23.72
		PNAT [58]	Positional predictor, model the position of target tokens	ARXIV 2019	1	7.3x	23.05*
		SynST [59]	Parse decoder, autoregressively predict a chunked parse tree	ACL 2019	$N/6$	4.9x	20.74
		LaNAT [60]	Delta Posterior, continuous latent variables	AAAI 2020	1	6.8x	24.20
		Reorder-NAT [61]	Reordered the source sentence into a pseudo-translation	AAAI 2021	1	16.1x	22.79
		AligNART [62]	Aligner module, alignment decomposition strategy	EMNLP 2021	1	13.4x	26.40
		CNAT [63]	Categorical codes, without external syntactic parser	NAACL 2021	1	10.4x	25.56*
		SNAT [64]	Incorporate the explicit syntactic and semantic structures	EACL 2021	1	22.6x	24.64*
		Fully-NAT [38]	Several tricks to improve the Fully NAT	ACL# 2021	1	16.5x	27.49*
	Other enhancements -based methods	ENAT [23]	Phrase-table lookup, embedding mapping	AAAI 2019	1	25.3x	20.65
		NAT-REG [22]	Similarity and reconstruction regularization	AAAI 2019	1	27.6x	20.65
		LAVA NAT [65]	Vocabulary attention, reorder prediction labels of a word	ARXIV 2020	1	29.3x	25.72
		CCAN [66]	Context-aware cross-attention, local and global contexts	COLING 2020	10	-	27.50
		DSLP [67]	Deep supervision, additional layer-wise predictions	ARXIV 2021	1	14.8x	27.02
		DAD [68]	Decoder Input Transformation, backward dependency modeling	ARXIV 2022	1	14.7x	27.51
Criterion	Loss function	CTC [36]	Compute and stores partial log-probability	EMNLP 2020	1	18.7x	25.60
		BoN-Joint [69]	N-gram level loss, minimize the Bag-of-Ngrams difference	AAAI 2020	1	10.8x	20.90
		AXE-NAT [70]	Aligned cross-entropy, a differentiable dynamic program	ICML 2020	1	15.3x	23.53*
		EISL [71]	Compute the n-gram matching different, more rubust	ARXIV 2021	1	-	24.17*
		OAXE-NAT [39]	Order-agnostic cross-entrop, hungarian algorithm	ICML 2021	1	15.3x	26.10*
Decoding	Semi-autoregressive decoding	Semi-NAT [17]	Generate muti-tokens at one decoding step	EMNLP 2018	$N/2$	1.5x	26.90
		RecoverSAT [72]	Recover segment, recover mistakens of muti-tokens	ACL 2020	$N/2$	2.2x	27.11
	Iterative decoding	Mask predicted [18]	Mask several tokens with the lowest probability scores	EMNLP 2019	10	1.7x	27.03
		Easy-first [34]	Easy-first order, update each position given easier tokens	ICML 2019	Adaptive	3.5x	27.34
		Insert [53]	Insert tokens during each iteration	ICML 2019	-	-	-
		Insert and delete [34]	Insert or delete tokens during each iteration	NeurIPS 2019	-	-	-
	Mixed decoding	Unify [35]	Unified approach, conditional permutation language modeling	COLING 2020	10	-	26.35
		Diformer [73]	Directional transformer, directional embedding and self-attention	ARXIV 2021	10	-	27.99
Benefiting from Pre-trained Models	AT models	Imitate-NAT [74]	Imitation learning framework with imitate module	ACL 2019	1	18.6x	22.44*
		Hint-NART [75]	Hints from the hidden state, constrain attention distributions	EMNLP 2019	1	30.2x	21.11
		ENGINE [76]	Energy-based inference, minimize the AT model's energy	ACL 2020	-	-	-
		EM+ODD [77]	Unified framework, dynamically optimize AT and NAT	ICML 2020	1	16.4x	24.54
		FCL-NAT [24]	Curriculum learning from better-trained state of AT model	AAAI 2020	1	28.9x	21.70
		MULTI-TASK NAT [78]	Shared encoder, dynamically mix two training loss	NAACL 2021	10	-	27.98*
		TCT-NAT [79]	Task-level curriculum learning, from AT to SAT, then to NAT	IJCAI 2021	1	27.6x	21.94
	Pre-trained language models	AB-Net [80]	Take two different BERT models as the encoder and decoder	NeurIPS 2020	-	2.4x	28.69*
		Bert+CRF-NAT [81]	Employ bert as a backbone, add a CRF Layer	EACL 2021	-	-	-
		CeMAT [40]	Aligned code-switching and masking, dynamic dual-masking	ARXIV 2022	10	-	27.20

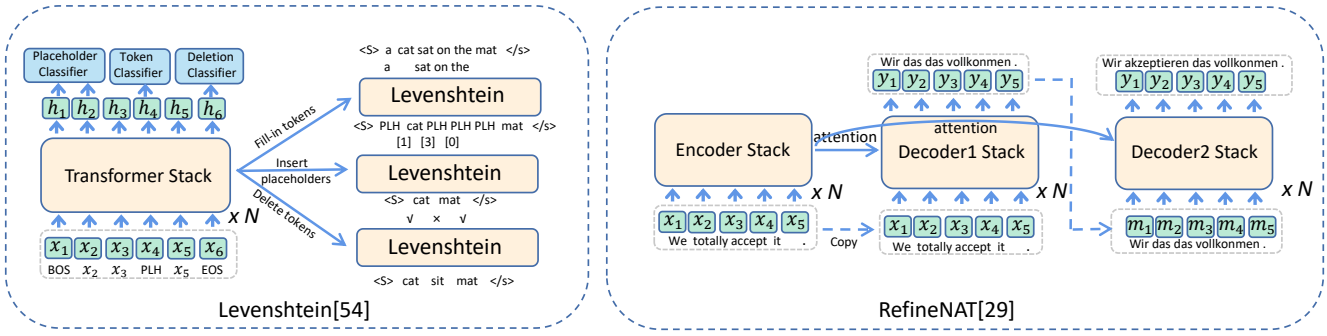
this glancing sampling strategy to a variable-based model. Song *et al.* [83] combine this glancing sampling strategy with a code-switch method for the task of multilingual machine translation. Ding *et al.* [51] divide training data into multiple granularities, such as words, phrases, and sentences, and propose a progressive multi-granularity training strategy to train the model from easy to hard. Apart from curriculum learning, consistency training is an effective method for autoregressive NMT models [84]. For NAT models, Xie *et al.* [52] utilize consistency training to improve the training consistency on different masked sentences. They assumed that the prediction of the same masked position should be consistent in different contexts or with different models. A similar idea is also explored for variational autoencoder-based latent-variable NAT models in recent papers [85], which propose posterior consistency regularization to improve the ability of models. They first apply data augmentation on both source and target sentences twice and then predict the latent variable and regularize these two results. Besides, contrastive learning is also adopted to improve the performance of NAT models [86], which optimizes the similarity of several different representations of the same token in the same sentence, resulting in more informative and robust representations.

4 MODELING

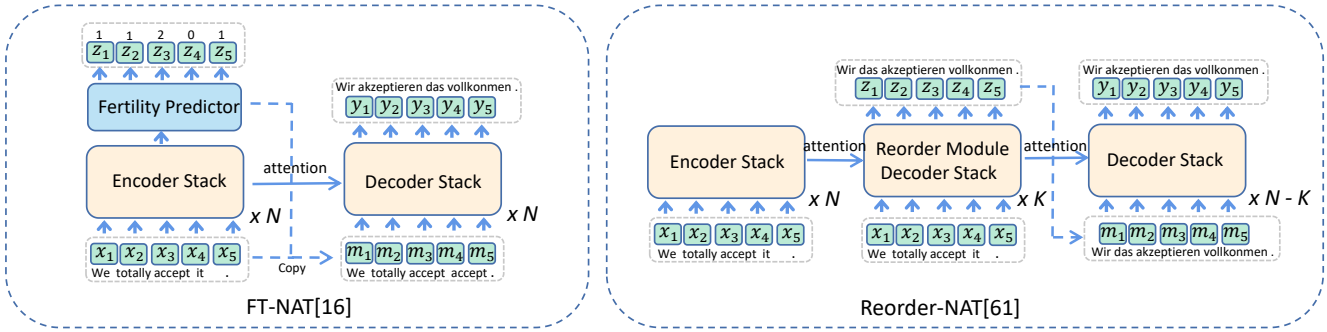
The model structure plays a critical role for NAT models to better capture the target side dependency. This section first introduces two popular model frameworks for NAT models: iteration-based methods and latent variable-based methods, then we summarize the efforts made on other enhancements-based methods for NAT models. The introduced methods are listed in the "Modeling" category of Table 1, and Figure 5 presents a few representative modeling methods.

4.1 Iteration-Based Methods

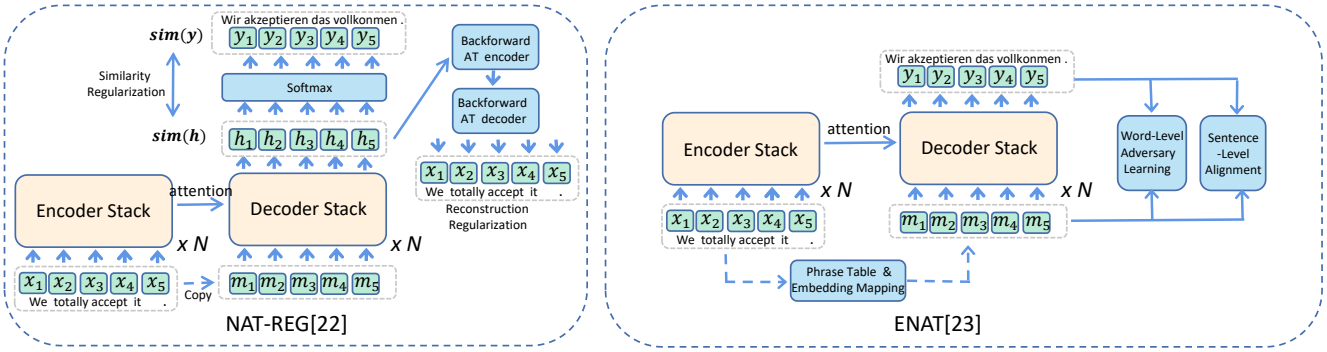
Iteration-based methods aim to find the trade-off between translation speed and quality. Instead of generating all target tokens in one pass, they learn the conditional distribution over partially observed generated tokens. Lee *et al.* [29] first propose the iterative model, and they utilize either the output of the previous iteration or the noised target sentence to



(a) An illustration of two iteration-based methods introduced in Section 4.1.



(b) An illustration of two latent variable-based methods introduced in Section 4.2.



(c) An illustration of two enhancements-based methods introduced in Section 4.3.

Fig. 5. We show the model structures of two iteration-based methods, e.g., two iteration-based methods, RefineNAT [29] and Levenshtein [54]; two latent variable-based methods, e.g., FT-NAT [16] and Reorder-NAT [61], where the fertility predictor and reorder module are applied to predict the latent variables; and two enhancements-based model, NAT-REG [22] and ENAT [23], and the corresponding enhancement module is also given.

initial the decoder input for refinements. Besides, refinements can be made upon the following operations:

Insertion and Deletion. Stern *et al.* [53] propose a model based on the insertion operation, which generates target tokens in the order of a balanced binary tree. When generating, the central words are generated first. Then in the subsequent iterations, the left and right sub-trees produce the words in their respective central positions. Besides, Gu *et al.* [54] further present the deletion operation.

Mask and Predict. Another line of work leverages the success of the masked language model, which is originally proposed by BERT [33]. Ghazvininejad *et al.* [18] extend it to the conditional masked language model (CMLM) by masking and predicting target tokens during training. During inference, in each iteration, a fraction of target tokens with low prediction probability will be masked and fed to the decoder for the next iteration. Based on this model, several follow-up works are proposed by: (1) jointly masking tokens [55], where the tokens in the source sentences are also masked; (2) introducing self-review mechanism [87], which applies an AR-decoder to help infuse sequential information; (3) selecting masked tokens with advanced strategies [26], [34], [88]. Geng *et al.* [26] focus on the importance of determining the tokens replaced by `[mask]` tokens in next iteration and propose a revisor and locator for rewriting. Kreutzer *et al.* [88] also explore the strategies to mask the suitable tokens in each iteration. Kasai *et al.* [34] mask and predict target tokens in an easy-first order instead of randomly.

4.2 Latent Variable-Based Methods.

Utilizing latent variables as part of the model is also a popular method to reduce the target side dependency. Latent

variable models maximize the following likelihood:

$$\mathcal{L}_{\text{Lat}} = \sum_{t=1}^{T} \log p(Z|X;\theta)p(y_t|Z,X;\theta),$$

where Z is a specific latent variable. The latent variable-based NAT models first predict a latent variable sequence, where each variable may be a chunk of words or include some other prompt information. Existing works mainly apply latent variables to capture the following information.

Prior Target Linguistic Information. Ma *et al.* [57] utilize a powerful mathematical framework called generative flow. Variational auto-encoders(VAE) based methods are also applied to model the dependency [89]. Shu *et al.* [60] and Lee *et al.* [47] model the latent variables as spherical Gaussian for every token in the encoder. Bao *et al.* [50] utilize a glancing sampling strategy to optimize latent variables.

Alignments between Source and Target Sentences. Gu *et al.* [16] pre-define the latent variable Z as fertility and use it to determine how many target words every source word is aligned to. Song *et al.* [62] predict the alignment by an aligner module as the latent variable Z.

Position Information of Target Tokens. Bao *et al.* [58] propose PNAT, which depends on the part of an extra positional predictor module to achieve the permutation Z. Ran *et al.* [61] propose ReorderNAT, a novel NAT framework that reorders the source sentence by the target word order to help the decision of word positions.

Syntactic Information of Target Sentence. Syntactic labels represent the sentence structure, which can be utilized to guide the generation and arrangement of target tokens. Akoury *et al.* [59] first introduce syntactic labels as a supervision to help the learning of discrete latent variables. However, the method needs an external syntactic parser to produce the syntactic reference tree, which is effective only in limited scenarios. To release the limitation, Bao *et al.* [63] propose to learn a set of latent codes that act like the syntactic label. Liu *et al.* [64] incorporate the explicit syntactic and semantic structures to improve the ability of NAT models. Specifically, they utilize Part of Speech (POS) and Named Entity Recognition (NER) to introduce these information.

4.3 Other Enhancements-based Methods

In addition to the above two popular frameworks for NAT models, many efforts have been made to improve the ability of capturing the target side dependency for NAT models at different stages, and the corresponding module is also added to their models. We summarize these methods into the following categories.

Enhancing the Input of Decoder. Since copying the source sentence to initial the decoder cannot offer any target information [16], Guo *et al.* [23] propose phrase-table lookup and embedding mapping methods to enhance the input of the decoder, which can feed tokens with some target information into the decoder, then help model learn the training data better. While the used phrase table is trained in advance, embedding mapping drew lessons on the idea of adversarial training and can perform word-level constraints to close the input and target sentence. Zhan *et al.* [68] also focus on the input of the decoder. They propose decoder input transformation, which transforms the decoder input into the target space. Then this can close the input and target-side embedding and help capture the target side dependency.

Supervising the Intermediate States. Several works give extra guidance to the decoder module. Firstly, additional attention modules are applied to learn more information. Li *et al.* [65] propose the Vocabulary Attention (VA) mechanism along with the Look-Around (LA) strategy to help the model capture long-term token dependencies of the target sentence. Ding *et al.* [66] propose a context-aware cross-attention module that focuses on both local and global contexts simultaneously and therefore enhances the supervision signal of neighbor tokens as well as the information provided by the source texts. Besides, Huang *et al.* [67] provide layer-wise supervision to the intermediate states of each decoder layer.

Improving the Output of Decoder. For the output of the decoder, Wang *et al.* [22] regularize the learning of the decoder representations by introducing similarity and reconstruction regularizations, where the former aims at avoiding similar hidden states to alleviate the repetitive translation problem, and the latter constraints the results to help address the problem of incomplete translations. Besides, Ran *et al.* [72] propose the RecoverSAT model to recover from repetitive and missing token errors by dynamically determining the length of segments that need to recover and then deleting repetitive segments.

5 Criterion

In addition to training data and model structure, training criterion is always another decisive factor for the success of neural network models. Most NMT models apply cross-entropy (CE) loss as their training criterion, which is calculated similarly to Equation 2.1:

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T} \log P(y_t|X;\theta),$$

where each $P(y_t|X;\theta)$ is calculated conditional independently by the NAT model with parameters θ. However, several researchers have pointed out that the CE loss may not be optimal for NAT models, and they propose better criteria to improve the performance of NAT models. This section compares these criteria with traditional CE loss, emphasizes their advantages, and summarizes them into the following categories.

Connectionist Temporal Classification (CTC). CTC based criteria [27] compute and store partial log-probability summations for all prefixes and suffixes of the output sequence by dynamic programming to alleviate the misalignment problem. Libovicky *et al.* [90] and Shu *et al.* [60] also use CTC loss to marginalize all the monotonic alignments between target and predictions, which can be written as

$$\mathcal{L}_{\text{CTC}} = -\sum_{a\in\beta(y)} \prod_{i} p(a_i|x,\theta)),$$

where a is a possible latent alignment, $\beta(y)$ denotes all possible alignments based on the CTC format.

N-Gram Based. N-gram based criteria [69] focus on n-gram level relationships. The word-level CE loss encourages NAT to generate the target tokens without considering the global correctness, which aggravates the weakness in capturing

target side dependency. Shao *et al.* [69] propose an n-gram level loss function to minimize the Bag-of-Ngrams (BoN) difference between the model output and the reference sentence. Guo *et al.* [55] also introduce the n-gram based dependency of target tokens to alleviate the problem of repetitive translations. N-gram based loss can be written as:

$$\mathcal{L}_{\text{BoN}} = \frac{\text{BoN-}L1}{2(T-n+1)},$$

where BoN-$L1$ is the L1 distance between the number of n-grams predicted by the NAT model and that in the reference sentence, which can be calculated as:

$$\text{BoN-}L1 = \sum_{g} |\text{BoN}_{\theta}(g) - \text{BoN}_{Y}(g)|,$$

where $g = (g_1, g_2, ..., g_n)$ is a possible n-gram set. $\text{BoN}_Y(g) = \sum_{t=0}^{T-n} 1\{y_{t+1:t+n} = g\}$ is the number of occurrences of g in sentence Y. $\text{BoN}_\theta(g)$ denotes the BoN for a NAT model with parameters α, which can be written as:

$$\begin{aligned}\text{BoN}_{\theta}(g) &= \sum_{Y} P(Y|X,\theta) * \text{BoN}_{Y}(g) \\ &= \sum_{Y} P(Y|X,\theta) * \sum_{t=0}^{T-n} 1\{y_{t+1:t+n} = g\} \\ &= \sum_{t=0}^{T-n} \prod_{i=1}^{n} P(y_{t+i=g_i|X,\theta})\end{aligned}$$

where X and Y denote the source and target sentences, respectively. Liu *et al.* [71] propose a novel Edit-Invariant Sequence Loss (EISL) which focuses on the n-gram matching to make the model perform more robustly when encountering inconsistent sequence order of source and target. They show that NAT benefits from this loss since the vanilla NAT model is struggling to model flexible generation order.

Order-Based. CE loss is sensitive to any inconsistent alignments between the prediction and target, which leads to penalizing a reasonable translation if it only mismatches the positions of target tokens. To soften the penalty for word order errors, Ghazvininejad *et al.* [70] propose aligned cross-entropy (AXE) loss, which uses a differentiable dynamic programming method to determine loss based on the best possible monotonic alignment between the ground-truth and the model predictions. The AXE loss is calculated as:

$$\mathcal{L}_{\text{AXE}} = -\sum_{t=1}^{T} \log P_{\alpha}(y_t|X;\theta) - \sum_{k \notin \theta} P_k(\epsilon),$$

where the first term indicates the aligned cross-entropy loss function between the target tokens and predictions, and the second term penalizes the unaligned predictions. Besides, Du *et al.* [39] further propose the order-agnostic cross-entropy (OAXE) loss, which applies the Hungarian algorithm to find the best possible alignment. The OAXE loss almost removes the penalty for order errors and guides NAT models to focus on lexical matching.

Given a parallel training sample (X, Y), we can define the alignment between a model prediction $\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_{T_{\hat{Y}}}\}$ and a target sentence $Y = \{y_1, y_2, ..., y_{T_Y}\}$ as an ordering of the set of target tokens Y, e.g., $O^i = \{y_{T_Y}, y_1, ..., y_{T_Y-1}\}$ denotes that tokens $\hat{y}_1, \hat{y}_2, ..., \hat{y}_{T_{\hat{Y}}}$ in model prediction $\hat{Y}$ are aligned with tokens $y_{T_Y}, y_1, ..., y_{T_Y-1}$ in target sentence Y respectively. Note that during training, $T_Y = T_{\hat{Y}}$. For each target sentence, we can get $T_Y!$ monotonic alignments. Based on each alignment state O^i, the corresponding CE loss can be calculated as:

$$\mathcal{L}_{\text{O}^i} = -\log P(O^i|X;\theta).$$

Given all possible alignment states $O = \{O^1, O^2, ..., O^{T_Y!}\}$, the OAXE objective is defined as finding the best alignment O^i to minimize the CE loss:

$$\mathcal{L}_{\text{OAXE}} = \arg\min_{O^i \in \mathbf{O}} (\mathcal{L}_{\text{O}^i})$$

where $-\log P(O^i|X;\theta)$ indicates the CE loss with ordering O^i. The introduced methods in this section are listed in the "Criterion" category of Table 1 and exemplified in Figure 6.

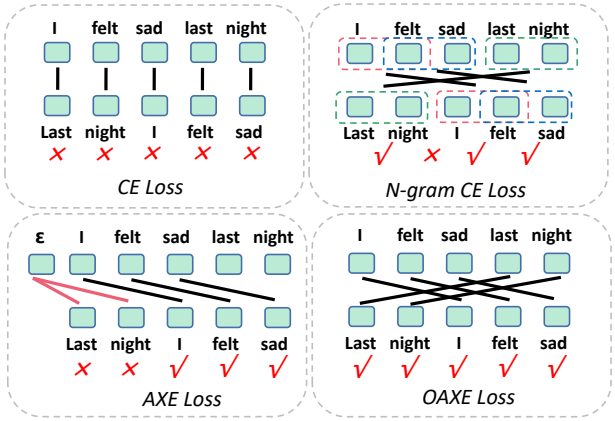


Fig. 6. An illustration of different loss functions. e.g., Model prediction: **Last night I feel sad** and Ground-truth: **I feel sad last night**. Traditional CE loss will give a penalty to all tokens. N-gram CE loss only finds a two-gram **night I** unreasonable. AXE loss finds the best possible monotonic alignment and penalizes unaligned tokens, denoted as ϵ, while OAXE loss removes the order errors and give no penalty to this prediction.

6 Decoding

The decoding stage is also crucial for neural machine translation models. Some works try to improve the NAT decoding schedule by applying different tricks. As mentioned in section 2.1, NAT models need to know the target length to guide decoding. And after the length is predicted, different decoding schedules are adopted to improve decoding. In this section, we will introduce various length prediction methods and decoding strategies.

6.1 Length Prediction

In AT models, the beginning and end of decoding are controlled by special tokens, including `[BOS]` (beginning of a sentence) and `[EOS]` (end of a sentence), which implicitly determine the target length during decoding. However, as all target tokens are generated in parallel in NAT models, there is no such special token or target information to guide the termination of decoding. NAT models must know the target length in advance and then generate the content based on it. Therefore, how to predict the correct length of the target

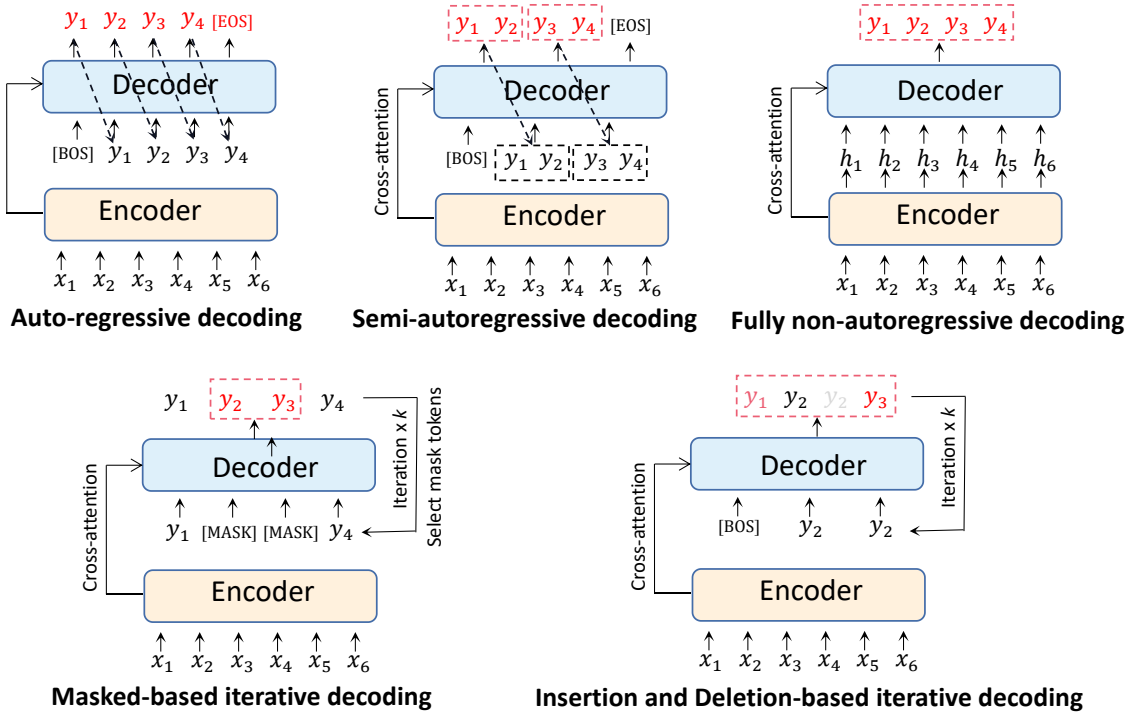


Fig. 7. An illustration of several different decoding strategies introduced in Section 6.

sentence is critical for NAT models. Different methods for target length prediction have been proposed.

Length Prediction Mechanism. Length information of target sentence is essential to NAT models as mentioned above. Gu *et al.* [16] propose a fertility predictor to decide how many times the source token will be copied when constructing the decoder input. Then, the sum of fertility numbers could be viewed as the length of the target sentence. Other length prediction methods are also proposed: (1) Classification modeling, which formulates the length prediction as a classification task and utilizes the encoder output to predict the target length or the length difference between the source and target [29], [74]; (2) Linear modeling, Sun *et al.* [25] try to use a linear function such as $T_y = \alpha\, T_x + B$ to directly calculate the target length based on source length; (3) Special token modeling by introducing a special `[LENGTH]` token [18], [37], [76]. Akin to the `[CLS]` token in BERT, the `[LENGTH]` token is usually appended to the encoder input, and the model is then trained to predict the length of the target sentence utilizing the hidden output of the `[LENGTH]` token; (4) CTC-based modeling, several models implicitly determine the target length from the word alignment information [36], [90] based on the connectionist temporal classification (CTC) [27] results.

Length Prediction Improvements. Inevitably, there is a deviation between the predicted length and the true length. To release the inherent uncertainty of the data itself, length parallel decoding (LPD) [23], [29] and noise parallel decoding (NPD) [16], [18] are widely utilized during inference. (1) LPD is often used in classification-based models. Once the length T is determined, they choose an LPD window m and then obtain multiple translation results with lengths in the range $[T-m, T+m]$. A pre-trained auto-regressive model is then used to score and select the best overall translation. (2) Models that adopt NPD choose the top m lengths with the highest length prediction probability and return the translation candidate with the highest log probabilities on the average of all tokens.

6.2 Decoding Strategy

Fully NAT models adopt only one-step decoding, which can greatly speed up decoding but fail to achieve high-quality translation. As shown in Table 1, the performance of iterative-based models is generally better than that of fully NAT models, indicating that NAT models fail to capture the target side dependency correctly with only one-step decoding. Figure 7 depicts several typical decoding strategies.

Semi-Autoregressive Decoding. Semi-autoregressive decoding is adopted for SAT models, which generates multiple target tokens at one decoding step. This decoding manner does not remove the dependency of target tokens completely. Several methods are proposed based on the semi-autoregressive decoding manner, such as: (1) Syntactic labels based [59], which applies a syntactic parser to produce the syntactic reference tree for the tokens in the current decoding step, then a group of tokens with a close syntactic relationship will be generated at one step. (2) Recover mechanism [72], which aims to alleviate the multi-modality problem by introducing a recovered segment. Once a group of tokens is generated, the model will recover from missing and repetitive token errors. (3) Aggressive decoding [91], which first aggressively decodes several tokens as a draft in a non-autoregressive manner and then verifies them in an auto-regressive manner. This method can improve the translation quality and lower the latency as the drafting and verification can execute in parallel.

Iterative Decoding. The iterative decoding manner is adopted for iterative NAT models, which provides partial

target information on each decoding step. Different iterative models mentioned above usually have their decoding strategies and the termination condition, including: (1) Mask prediction algorithm [18], which generates an entire sequence in parallel within a constant number of cycles, a specific number of tokens with low confidence would be masked on each decoding step and re-generated on the next step until the iteration met a pre-determined number; (2) Mask prediction with easy-first policy [34], which modifies the mask prediction algorithm by updating each position with an easy to hard order given the prediction probability of previous iterations; (3) Insertion and deletion [53], [54], [92], which aims to generate target tokens in the order of a balanced binary tree. This is specially designed for iteration-based models mentioned in section 4.1. (4) Rewriting mechanism [26], which aims to rewrite the erroneous translation pieces on each decoding step, and a dynamic termination method is also applied. Besides, inspired by the beam search algorithm for the CTC-based model, Kasner *et al.* [93] also apply beam search and employ additional features in its scoring model to improve the fluency of NAT.

Mixed Decoding. Since different types of decoding strategies have been proposed for NAT, several works aim to combine these decoding strategies into a unified model [35], [73]. Tian *et al.* [35] propose a unified approach for machine translation that supports autoregressive, semi-autoregressive, and iterative decoding methods. Once the model is trained, any of the above decoding strategies can be applied by repeatedly determining positions and generating tokens on them. Taking a step further, Wang *et al.* [73] propose a directional Transformer, which models the AR and NAR generation with a unified framework by designing a special attention module. Their model supports four decoding strategies and can dynamically select strategies during each iteration. We provide an illustration of different decoding strategies in Figure 7, in which more details of these strategies with examples are shown in Figure 11 of the Appendix.

7 Benefiting from Pre-trained Models

To improve the performance of NAT models, various methods are proposed to leverage the information from other strong models, such as their AT counterparts and large-scale pre-trained language models. We will introduce these methods in the following content.

7.1 AT Models

Due to the strong performance of AT models, leveraging AT models to help the NAT model training is appealing. But they differ in model structure and decoding strategy. Therefore, different techniques to benefit the NAT model from their AT counterparts are proposed:

Training with the Supervision of AT Models. Wei *et al.* [74] propose a novel imitation learning framework, introducing a better trained AT demonstrator to supervise each decoding state of the NAT model across different times so that the problem of huge search space can be alleviated. Li *et al.* [75] design two kinds of hints from the hidden representation level to regularize the KL-divergence of the encoder-decoder attention between the AT and NAT models, which can help

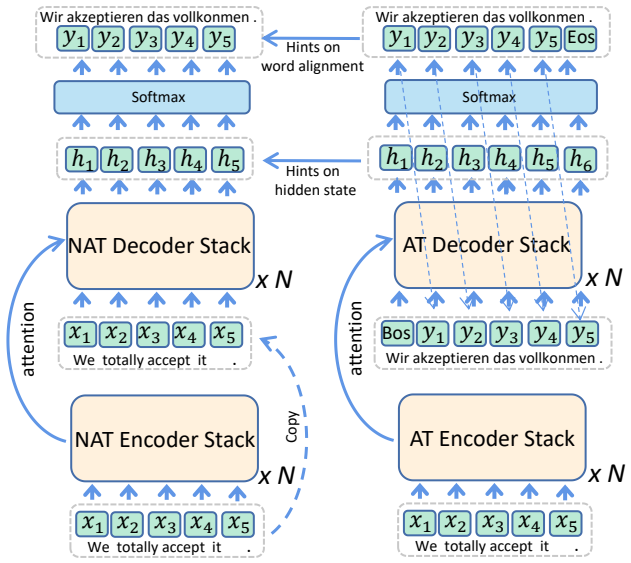


Fig. 8. An illustration of the model structure of Hint-NART [75] introduced in Section 7.1, which benefits from an AT model .

the training of NAT models, where its model structure is shown in Figure 8. Besides, Tu *et al.* [76] propose an energy-based inference network to minimize the energy of AT model and give several methods for relaxing the energy.

Fine-Tuning from AT Models. Guo *et al.* [24] utilize curriculum learning to fine-tune from a better-trained state of AT models, and two curricula for the decoder input and mask attention are applied. In addition, Liu *et al.* [79] propose task-level curriculum learning to shift the training strategy from AT to SAT gradually, and finally to NAT.

Training with AT Models Together. Sun *et al.* [77] propose an unified Expectation-Maximization (EM) framework. It optimizes both AT and NAT models jointly, which iteratively updates the AT model based on the output of the NAT model and trains the NAT model with the new output of AT model. Besides, Hao *et al.* [78] propose a model with a shared encoder and separated decoders for AT and NAT models. The training for these two models is controlled by different weights to mix two training losses.

7.2 Pre-Trained Language Models

While large-scale pre-trained language models have been proven effective in auto-regressive machine translation [94], [95], efforts are also made for non-autoregressive machine translation. Guo *et al.* [80] incorporate BERT into machine translation based on the mask-predict decoding method, which initializes the encoder and decoder with corresponding pre-trained BERT models, and inserts adapter layers into each layer. Su *et al.* [81] employ BERT as the backbone model and add a CRF output layer for better capturing the target side dependency to improve the performance further. Li *et al.* [40] propose a conditional masked language model with an aligned code-switching masking strategy to enhance the cross-lingual ability. The proposed model can be fine-tuned on both NAT and AT tasks with promising performance.

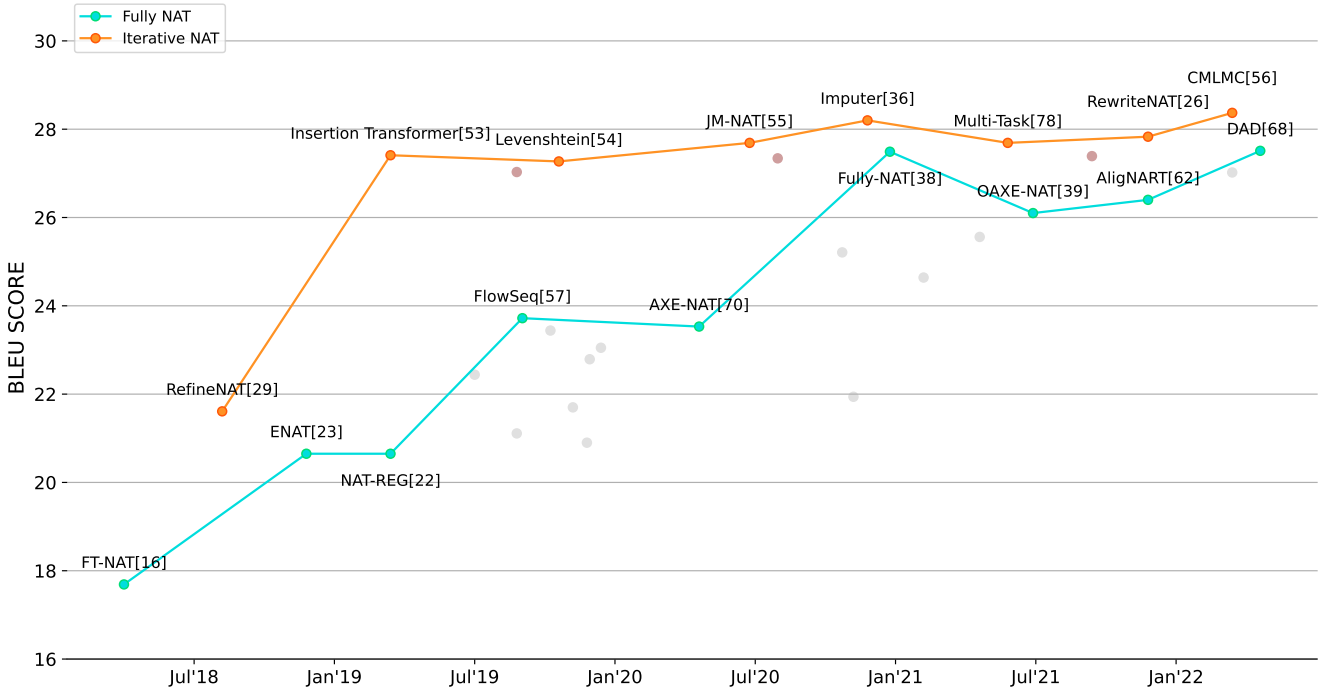


Fig. 9. The evolution of BLEU scores on WMT14 EN→De translation by the time of Fully NAT and Iterative NAT. Note that the performance of iterative NAT models is commonly better than fully NAT models at different stages, but the gap is narrowing with the development of NAT models.

8 Summary of Non-Autoregressive NMT

All of the above techniques can mitigate the challenge of failing to capture the target side dependency more or less by reducing the reliance of NAT models on target tokens. Since NAT models are essentially data-driven, their performance highly depends on data volume, quality, and learning strategies. Thus, data manipulation methods are almost indispensable for existing NAT works. Various KD methods can reduce the complexity of the training corpus, while data learning strategies can facilitate the understanding and learning of training data. Another critical element in capturing the target side dependency is NAT model structure, e.g., iteration-based methods, latent variables, and various add-ons for the decoder module. Another critical element in capturing the target side dependency is the NAT model structure, e.g., iteration-based methods, latent variables, and various add-ons for the decoder module. In addition to data manipulation and model structure, better training criteria are proposed to make up for the deficiency of cross-entropy loss, e.g., leveraging CTC-based criteria to alleviate the misalignment problem, introducing n-gram-based criteria to capture global context other than word-level correctness, and designing order-based criteria to soften the penalty for reasonable translations but with mismatched tokens at the target positions. Since the differences between AT and NAT models are mainly manifested in the decoder part, different improving skills for the NAT decoding mode are also presented. Typical strategies include performing target length prediction to guide the end of decoding and improving the one-step decoding by keeping part of the target side dependency information in semi-autoregressive decoding, providing partial target information in iterative decoding, and exploring their combinations in mixed decoding. Besides, leveraging the information from other strong models can further improve the performance of NAT models, such as utilizing information from their AT counterparts and large-scale pre-trained language models.

To help researchers and engineers select appropriate techniques in applications, we also conduct a brief comparison between existing methods on their effectiveness and inference speed. As shown in Figure 9, iterative-based NAT models generally achieve higher BLEU scores than fully NAT methods at the cost of multiple inference time, but their performance gap is rapidly shrinking, e.g., the recent combination of CTC length prediction, latent variable, and extra upsampling module can achieve competitive performance with strong iterative-based NAT methods. It can be expected that fully NAT methods can achieve better performance while maintaining their speed advantage with emerging effective strategies and a suitable combination. Figure 10 reports the correlations between performance and inference speed achieved by representative NAT methods. Methods in the lower left part of Figure 10, e.g., Disco [34], FT-NAT [16] can achieve much faster inference speed but at the cost of significant performance decrease, while methods in the upper right part can make a better trade-off between speed-up and performance. A few powerful NAT methods can even achieve comparable and slightly better performance than the strong AT model with a speed advantage.

9 Extensive Applications Beyond NMT

After seeing the success of non-autoregressive (NAR) techniques on neural machine translation, these strategies are also widely applied to extensive text generation tasks, semantic parsing [96], [97], text to speech [98], [99], etc. In this section, we will conduct a brief discussion about these works.

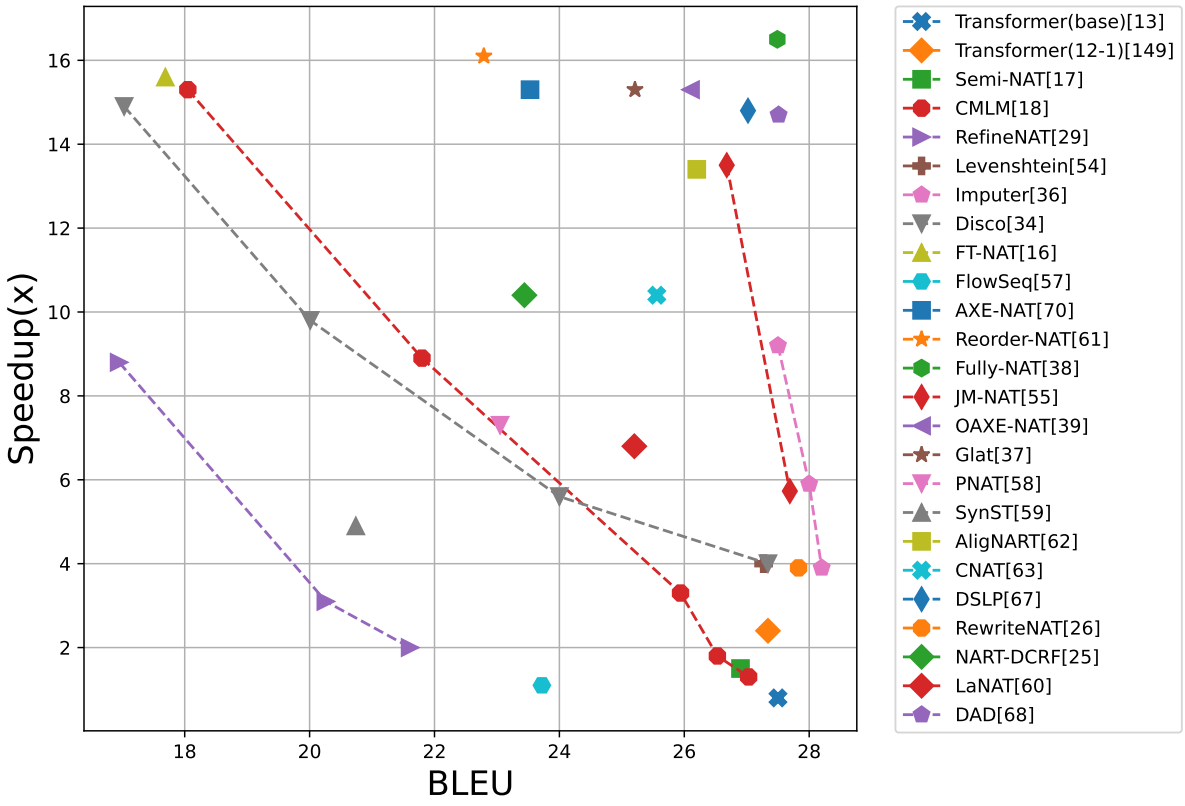


Fig. 10. BLEU v.s. Speedup. BLEU score is reported on the WMT14 EN→DE test set. Speedup is reported for NAT models compared with corresponding AT models. Dotted lines denote the different scores of iterative models achieved with different iterations. Note that the ideal model should appear in the top right-hand corner.

9.1 Text Generation

The inference efficiency is not only required for neural machine translation but also indispensable for many other text generation tasks [81], [100], [101]. Existing works of introducing NAR techniques into text generation tasks focus on automatic speech recognition [102], [103], [104], text summarization [105], grammatical error correction [106], [107], dialogue [108], [109]. Resembling the encountered challenge of NAT models in Section 2.2, representative works of non-autoregressive text generation mainly address the problems of missing target-side information and length prediction. According to the involved tasks, we structure these works into different groups, including general-purpose NAR methods and typical models for each specific generation task.

General-Purpose NAR Text Generation. Some works aim to design a general NAR method that can support multiple text generation tasks. Su *et al.* [81] employ BERT as the backbone of a NAR generation model for machine translation, sentence compression, and summarization. They add a CRF output layer on the BERT architecture for non-autoregressive tasks. For length prediction, they adopt two special tokens `[eos]` to dynamically guide the end of the generation. They extend the architecture of BERT to capture the target side dependency better and improve the performance further. For length prediction, they propose a simple and elegant decoding mechanism to help the model determine the target length on-the-fly. Jiang *et al.* [101] propose MIx Source and pseudo-Target (MIST), which adopts an iterative training mechanism to improve the ability of the NAR model without introducing extra cost during inference for question generation, summarization, and paraphrasing tasks. Jiang *et al.* [101] propose a new paradigm to adopt pre-trained encoders for NAR text generation tasks. They propose a simple and effective iterative training method, MIx Source and pseudo Target (MIST), for the training stage without introducing extra cost during inference. Yang *et al.* [100] attempt to explore the alternatives for KD in text summarization and story generation. They focus on linguistic structure predicted by a Part-of-Speech (POS) predictor to help alleviate the multimodality problem. Qi *et al.* [105] explore to design a large-scale pre-trained model that can support different decoding strategies when applied to downstream tasks. Concretely, they leverage different attention mechanisms during the training stage and fine-tuning strategies to adapt from AR to NAR generation. To verify the effectiveness of their model, they evaluate the proposed method for question generation, summarization, and dialogue generation tasks. Agrawal *et al.* [110] propose a framework that adopts an imitation learning algorithm for applying NAR models to

editing tasks such as controllable text simplification and abstractive summarization. They introduce a roll-in policy and a controllable curriculum to alleviate the mismatching problem between training and inference.

Task-Specific NAR Text Generation. Many other works introduce NAR methods for a specific text generation task.

- ***Automatic Speech Recognition.*** Consistent with neural machine translation, automatic speech recognition (ASR) has benefited dramatically from non-autoregressive models. NAR ASR models can significantly speed up the decoding process but also suffer from lower recognition accuracy due to the failure of capturing target side dependency. The difference reflects in the processing unit, which is a unique characteristic in NAR ASR [102]. The models with token-level processing units need length prediction, while models with frame-level need not. Thus, many NAR methods in neural machine translation cannot be directly used for ASR, but require specific modifications and designs, e.g., Iterative refinement-based [111], [112], Audio-CMLM [113], Imputer [114], Mask-CTC [115], and Insertion-based [53], [92] methods. Considering that the most widely used CTC method in NAR ASR is under the assumption that there exists strong conditional independence between different token frame predictions, researchers have made considerable efforts to optimize the vanilla CTC-based model [26], [104], [116], [117], [118], [119], [120]. Simultaneously, similar to the NAT method, the NAR ASR model can also benefit from pre-trained models, e.g., BERT [103], [121]. Besides, Higuchi *et al.* [102] carry out a comparative study on NAR ASR to better understand this task.
- ***Summarization.*** The summarization task is less subject to target-side dependency modeling than neural machine translation since all the target output information is explicitly or implicitly included in the long text input. As a result, NAR methods for the summarization task mainly alleviate the challenge of length prediction. For instance, a non-autoregressive unsupervised summarization (NAUS) model has been proposed recently [122], which first performs an edit-based search towards a heuristically defined score and then generates a summary as a pseudo-ground-truth. The authors also propose a length-control decoding approach for better target length prediction.
- ***Grammatical Error Correction.*** Grammatical Error Correction (GEC) is an important NLP task that can automatically detect and correct grammatical errors within a sentence. As most contents of a sentence are correct and unnecessary to be modified for the GEC task, the problem of lacking target-side information can be effectively alleviated. Thus, NAR methods are more feasible for this task. Li *et al.* [107] focus on the variable-length correction scenario for Chinese GEC. They employ BERT to initialize the encoder and add a CRF layer on the initialized encoder, augmented by a focal loss penalty strategy to capture the target side dependency. Besides, Straka *et al.* [106] propose a character-based non-autoregressive GEC approach for Czech, German and Russian languages, which focuses on sub-word errors.
- ***Dialogue.*** Dialogue generation has achieved remarkable progress in the last few years, and many methods have been proposed to alleviate the notorious problem of diversity [123]. However, due to their auto-regressive generation strategy, these dialogue generation models suffer from low inference efficiency for generating informative responses. Inspired by the advances made in NAT [25], [58], NAR models are adopted in dialogue generation to lower the inference latency, where the response length is predicted in advance. Han *et al.* [108] apply the NAR model to model the bidirectional conditional dependency between contexts (x) and responses (y). They also point out that NAR models can produce more diverse responses. Zou *et al.* [109] propose a concept-guided non-autoregressive method for open-domain response generation, which customizes the Insertion Transformer to complete response and then facilitates a controllable and coherent dialogue. These NAR models for dialogue generation can significantly improve response generation speed. Besides, NAR methods can improve task-oriented dialogue systems by enhancing the spoken language understanding sub-task [124].
- ***Text Style Transfer.*** Auto-regressive models have been widely used in unsupervised text style transfer. Despite their success, they suffer from high inference latency and low content preservation problems. Several works explore non-autoregressive (NAR) decoding to alleviate these problems. Ma *et al.* [125] first directly adapt the common training scheme from the AR counterpart in their NAR method and then propose to enhance the NAR decoding from three perspectives: knowledge distillation, contrastive learning, and iterative decoding. They also explore the potential reasons why these methods can narrow the performance gap with AR models. Huang *et al.* [126] point out that the auto-regressive manner might generate some irrelevant words with strong styles and ignore part of the source sentence content. They propose a NAR generator for unsupervised text style transfer (NAST), which effectively avoids irrelevant words by alignment information. NAST can dramatically improve transfer performance with efficient decoding speed.

9.2 Semantic Parsing

Compared with the non-autoregressive text generation tasks, non-autoregressive semantic parsing relies more on the length prediction mechanism, in which minor differences can lead to entirely different results. Several NAR models applied to semantic parsing are inspired by CMLM [18] but with better length prediction mechanisms. Babu *et al.* [96] study the potential limitations of the original CMLM when applied for semantic parsing and designed a new LightConv Pointer model to improve it, where the target length is computed by a separate module of multiple layers of CNNs with gated linear units. They also use label smoothing to avoid the easy over-fitting in length prediction. During inference, iterative refinement does not bring many benefits to task-oriented semantic parsing, and thus only one step is applied. Shrivastava *et al.* [97] design Span Pointer Networks based on CMLM with a span prediction mechanism to decide the target length. The length module of semantic parsing merely needs frame syntax to perform span prediction, while text generation requires both syntax and semantics.

9.3 Text to Speech

Significant progress has also been made in the non-autoregressive text to speech (NAR TTS) task. Ren *et*

al. [98] point out three main problems in autoregressive TTS compared with the non-autoregressive fashion, i.e., the speed of the inference stage is slow, the generated speech is not robust, and the generated speech is unable to be controlled. Accordingly, they present a model based on Transformer in a non-autoregressive manner to alleviate the above three problems. Besides, one-to-many (O2M) mapping problem is typical in NAR TTS since differences lie in human speaking greatly. Many other NAR TTS models are also proposed to alleviate this problem and improve speech quality. Peng *et al.* [99] propose ParaNet, which extracts attention from the auto-regressive TTS model and then re-defines the alignment. Lu *et al.* [127] apply the variational auto-encoder structure to model the alignment information with a latent variable and further use the attention-based soft alignment strategy. Shah *et al.* [128] propose a NAR model by replacing the attention module of the conventional attention-based TTS model with an external duration model for low-resource and highly expressive speech. Besides, a very deep VAE model with residual attention also benefits the NAR TTS [129]. Notice that the above models may need a teacher model to guide their learning. Lee *et al.* [130] propose a bidirectional inference variational auto-encoder to rely less on the teacher model and meanwhile without decreasing the performance. Since over-smoothing is a severe problem that harms the performance of NAR TTS models, many works focus on alleviating this problem. Ren *et al.* [131] summarize these methods into the two categories, i.e., simplify data distributions [132], [133], which provides more conditional input information, and enhance modeling methods [134], [135], which try to enhance the model capacity to fit the complex data distributions. Ren *et al.* [131] combine these two methods to improve the performance of NAR TTS further. The diversity problem of TTS is also explored in recent work. Bae *et al.* [136] propose a variational autoencoder with the hierarchical and multi-scale structure for NAR TTS (HiMuV-TTS) to improve the diversity of generated speech.

9.4 Speech Translation

Much progress has also been made in speech translation along with the development of NAR ASR models mentioned in section 9.1. Many NAR ASR models are applicable for end-to-end speech translation [137] by completing the automatic speech recognition and machine translation stages simultaneously. Since speech translation resembles text translation, effective strategies applied in text translation are also introduced to speech translation. In seeing the success of connectionist temporal classification (CTC) on machine translation [60], Chuang *et al.* [138] propose CTC-based speech-to-text translation model. They construct an auxiliary speech recognition task based on CTC to further improve performance. Inaguma *et al.* [139] propose Orthros to jointly train the NAR and AR decoders on a shared speech encoder, which is similar to sharing encoder structure in machine translation [78]. Besides, a rescoring mechanism is proposed for Orthros [140], in which an auxiliary shallow AR decoder is introduced to choose the best candidate. On the NAR side, they use CMLM and a CTC-based model as NAR decoders, denoted as Orthros-CMLM and Orthros-CTC, respectively. Such muti-decoder is also widely used for speech translation [141], [142], which is a two-pass decoding method that decomposes the overall task into two sub-tasks, i.e., ASR and machine translation. Inaguma *et al.* [143] propose Fast-MD, where the hidden intermediates are generated in a non-autoregressive manner by a Mask-CTC model. They also introduce a sampling prediction strategy to reduce the mismatched training and testing.

9.5 Others

In addition to the success of NAR methods on the above-mentioned tasks, many researchers have conducted a pilot study on other scenarios. Agrawal *et al.* [144] introduce a non-autoregressive approach for the task of controllable text simplification, where the model iteratively edits an input sequence and incorporates lexical complexity information into the refinement process to generate comparable simplifications. Information extraction (IE) also benefits from the non-autoregressive technique [145]. In essence, the facts in plain text are unordered, but the AR models need to predict the following fact conditioned on the previously decoded ones. Yu *et al.* [145] propose a novel non-autoregressive framework, named MacroIE, for OpenIE, which treats IE as a maximal clique discovery problem and predicts the fact set at once to relieve the burden of predicting fact order. For video generation (VG), Yu *et al.* [146] propose a dynamics-aware implicit generative adversarial network (DIGAN) for non-autoregressive video generation, which greatly increases inference speed via parallel computing of multiple frames. For voice conversion (VC), Hayashi *et al.* [147] extend the FastSpeech2 model in NAR TTS to the voice conversion task and introduce a convolution-augmented Transformer (Conformer). The proposed method can learn both local and global context information of the input sequence and extend variance predictors to variance converters to transpose the prosody components of the source speaker. Besides, self-supervised speech representations are effective in various speech applications. However, existing representation learning methods generally rely on the autoregressive model, leading to low inference efficiency. Liu *et al.* [148] propose Non-Autoregressive Predictive Coding (NPC) to learn speech representations in a non-autoregressive manner by only considering local dependencies of speech, which can significantly improve inference speed.

10 Conclusion and Outlooks

This paper reviews the development of non-autoregressive methods in neural machine translation and other related tasks. We first summarize the main challenge encountered in NAT research. Then, we structure existing solutions from different perspectives, including data manipulation, modeling, criterion, decoding, and benefiting from pre-trained models, along with a discussion on their effectiveness and inference speed. Besides, we present an overview of the applications of NAR methods in extensive tasks, e.g., summarization, semantic parsing, text to speech, and speech translation. We hope this survey can help researchers and engineers better understand the non-autoregressive techniques and choose suitable strategies for their application tasks.

Although impressive progress has been made on non-autoregressive models, there still exist some open problems:

- KD is the most effective method utilized in NAR models, which depends on pre-training an AR model in advance. However, how to release this condition and improve the performance of NAR models on raw datasets are worthy of further consideration.
- Iterative-based models achieve comparable performance with AR models, with extra computation cost. Recent works [149], [150] also show that their speedup w.r.t AR models will diminish when decoding with large batch size. Therefore, more attention should be paid to improving the performance of fully NAR models.
- Although existing length prediction strategies can achieve appealing performance on many tasks with easy-to-learn alignment patterns, the fixed target length damages their flexibility in generation, which may prevent the further application of NAR methods on extensive tasks, e.g., various open-ended generation tasks with a wide dynamic range for the target length. As a result, dynamic length prediction mechanisms are expected when introducing NAR methods to more tasks.
- AR models are generally applied to various application scenarios, including bilingual and multilingual, high-resource and low-resource, etc. However, most applications of NAR models are limited to the bilingual scenario until now. Therefore, to expand the impact of NAR models, it is worthy of applying NAR to more application scenarios.
- In recent years, considerable efforts have been made to enhance auto-regressive models with powerful pre-training techniques and models, with impressive performance being achieved. However, only very few papers apply these powerful pre-trained models to help NAR models [80], [101], and there is only a preliminary exploration of the pre-training techniques for NAR models [40], [105]. Thus, it is promising to explore pre-training methods for non-autoregressive generation and other related tasks.

REFERENCES

[1] H. Somers, "An introduction to machine translation," 1992.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[6] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[7] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[8] R. Sennrich, B. Haddow, and A. Birch, "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)*, 2016, pp. 1715–1725.

[9] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.

[10] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[11] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1243–1252.

[12] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[14] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," in *International Conference on Learning Representations*, 2018.

[15] L. Wu, Y. Wang, Y. Xia, F. Tian, F. Gao, T. Qin, J. Lai, and T.-Y. Liu, "Depth growing for neural machine translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5558–5563.

[16] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," in *ICLR*, 2018.

[17] C. Wang, J. Zhang, and H. Chen, "Semi-autoregressive neural machine translation," in *EMNLP*, 2018, pp. 479–488.

[18] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *EMNLP-IJCNLP*, 2019, pp. 6112–6121.

[19] W. Xu, S. Ma, D. Zhang, and M. Carpuat, "How does distilled data complexity impact the quality and confidence of non-autoregressive machine translation?" *arXiv preprint arXiv:2105.12900*, 2021.

[20] L. Ding, L. Wang, X. Liu, D. F. Wong, D. Tao, and Z. Tu, "Understanding and improving lexical choice in non-autoregressive translation," in *ICLR*, 2020.

[21] J. Guo, M. Wang, D. Wei, H. Shang, Y. Wang, Z. Li, Z. Yu, Z. Wu, Y. Chen, C. Su *et al.*, "Self-distillation mixup training for non-autoregressive neural machine translation," *arXiv preprint arXiv:2112.11640*, 2021.

[22] Y. Wang, F. Tian, D. He, T. Qin, C. Zhai, and T.-Y. Liu, "Non-autoregressive machine translation with auxiliary regularization," in *AAAI*, vol. 33, no. 01, 2019, pp. 5377–5384.

[23] J. Guo, X. Tan, D. He, T. Qin, L. Xu, and T.-Y. Liu, "Non-autoregressive neural machine translation with enhanced decoder input," in *AAAI*, vol. 33, no. 01, 2019, pp. 3723–3730.

[24] J. Guo, X. Tan, L. Xu, T. Qin, E. Chen, and T.-Y. Liu, "Fine-tuning by curriculum learning for non-autoregressive neural machine translation," in *AAAI*, vol. 34, no. 05, 2020, pp. 7839–7846.

[25] Z. Sun, Z. Li, H. Wang, D. He, Z. Lin, and Z. Deng, "Fast structured decoding for sequence models," *NeurIPS*, vol. 32, pp. 3016–3026, 2019.

[26] X. Geng, X. Feng, and B. Qin, "Learning to rewrite for non-autoregressive neural machine translation," in *EMNLP*, 2021, pp. 3297–3308.

[27] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006, pp. 369–376.

[28] J. F. Kolen and S. C. Kremer, *A field guide to dynamical recurrent networks*. John Wiley & Sons, 2001.

[29] J. Lee, E. Mansimov, and K. Cho, "Deterministic non-autoregressive neural sequence modeling by iterative refinement," in *EMNLP*, 2018, pp. 1173–1182.

[30] W. Zhang, Y. Feng, F. Meng, D. You, and Q. Liu, "Bridging the gap between training and inference for neural machine translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4334–4343.

[31] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[32] L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A study of reinforcement learning for neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3612–3621.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[34] J. Kasai, J. Cross, M. Ghazvininejad, and J. Gu, "Parallel machine translation with disentangled context transformer," *arXiv preprint arXiv:2001.05136*, 2020.

[35] C. Tian, Y. Wang, H. Cheng, Y. Lian, and Z. Zhang, "Train once, and decode as you like," in *COLING*, 2020, pp. 280–293.

[36] C. Saharia, W. Chan, S. Saxena, and M. Norouzi, "Non-autoregressive machine translation with latent alignments," in *EMNLP*, 2020, pp. 1098–1108.

[37] L. Qian, H. Zhou, Y. Bao, M. Wang, L. Qiu, W. Zhang, Y. Yu, and L. Li, "Glancing transformer for non-autoregressive neural machine translation," in *ACL-IJCNLP*, 2021, pp. 1993–2003.

[38] J. Gu and X. Kong, "Fully non-autoregressive neural machine translation: Tricks of the trade," in *Findings of ACL-IJCNLP*, 2021, pp. 120–133.

[39] C. Du, Z. Tu, and J. Jiang, "Order-agnostic cross entropy for non-autoregressive machine translation," in *ICML*. PMLR, 2021, pp. 2849–2859.

[40] P. Li, L. Li, M. Zhang, M. Wu, and Q. Liu, "Universal conditional masked language pre-training for neural machine translation," *ACL*, 2022.

[41] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[42] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *EMNLP*, 2016, pp. 1317–1327.

[43] C. Zhou, J. Gu, and G. Neubig, "Understanding knowledge distillation in non-autoregressive machine translation," in *ICLR*, 2019.

[44] Y. Ren, J. Liu, X. Tan, Z. Zhao, S. Zhao, and T.-Y. Liu, "A study of non-autoregressive model for sequence generation," in *ACL*, 2020, pp. 149–159.

[45] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1607–1616.

[46] T. Shen, M. Ott, M. Auli, and M. Ranzato, "Mixture models for diverse machine translation: Tricks of the trade," in *International conference on machine learning*. PMLR, 2019, pp. 5719–5728.

[47] J. Lee, D. Tran, O. Firat, and K. Cho, "On the discrepancy between density estimation and sequence generation," in *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, 2020, pp. 84–94.

[48] L. Ding, L. Wang, X. Liu, D. F. Wong, D. Tao, and z. Tu, "Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation," in *ACL-IJCNLP*, 2021, pp. 3431–3441.

[49] J. Zhou and P. Keung, "Improving non-autoregressive neural machine translation with monolingual data," in *ACL*, 2020, pp. 1893–1898.

[50] Y. Bao, H. Zhou, S. Huang, D. Wang, L. Qian, X. Dai, J. Chen, and L. Li, "*latent*-glat: Glancing at latent variables for parallel text generation," *ACL*, 2022.

[51] L. Ding, L. Wang, X. Liu, D. F. Wong, D. Tao, and Z. Tu, "Progressive multi-granularity training for non-autoregressive translation," in *Findings of ACL-IJCNLP*, 2021, pp. 2797–2803.

[52] P. Xie, Z. Li, and X. Hu, "Mvsr-nat: Multi-view subset regularization for non-autoregressive machine translation," *arXiv preprint arXiv:2108.08447*, 2021.

[53] M. Stern, W. Chan, J. Kiros, and J. Uszkoreit, "Insertion transformer: Flexible sequence generation via insertion operations," in *ICML*. PMLR, 2019, pp. 5976–5985.

[54] J. Gu, C. Wang, and J. Zhao, "Levenshtein transformer," *NeurIPS*, vol. 32, pp. 11 181–11 191, 2019.

[55] J. Guo, L. Xu, and E. Chen, "Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation," in *ACL*, 2020, pp. 376–385.

[56] X. S. Huang, F. Perez, and M. Volkovs, "Improving non-autoregressive translation models without distillation," in *ICLR*, 2022.

[57] X. Ma, C. Zhou, X. Li, G. Neubig, and E. Hovy, "Flowseq: Non-autoregressive conditional sequence generation with generative flow," in *EMNLP-IJCNLP*, 2019, pp. 4282–4292.

[58] Y. Bao, H. Zhou, J. Feng, M. Wang, S. Huang, J. Chen, and L. Li, "Non-autoregressive transformer by position learning," *arXiv preprint arXiv:1911.10677*, 2019.

[59] N. Akoury, K. Krishna, and M. Iyyer, "Syntactically supervised transformers for faster neural machine translation," in *ACL*, 2019, pp. 1269–1281.

[60] R. Shu, J. Lee, H. Nakayama, and K. Cho, "Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior," in *AAAI*, vol. 34, no. 05, 2020, pp. 8846–8853.

[61] Q. Ran, Y. Lin, P. Li, and J. Zhou, "Guiding non-autoregressive neural machine translation decoding with reordering information," in *AAAI*, vol. 35, no. 15, 2021, pp. 13 727–13 735.

[62] J. Song, S. Kim, and S. Yoon, "Alignart: Non-autoregressive neural machine translation by jointly learning to estimate alignment and translate," in *EMNLP*, 2021, pp. 1–14.

[63] Y. Bao, S. Huang, T. Xiao, D. Wang, X. Dai, and J. Chen, "Non-autoregressive translation by learning target categorical codes," in *NAACL-HLT*, 2021, pp. 5749–5759.

[64] Y. Liu, Y. Wan, J. Zhang, W. Zhao, and S. Y. Philip, "Enriching non-autoregressive transformer with syntactic and semantic structures for neural machine translation," in *EACL*, 2021, pp. 1235–1244.

[65] X. Li, Y. Meng, A. Yuan, F. Wu, and J. Li, "Lava nat: A non-autoregressive translation model with look-around decoding and vocabulary attention," *arXiv preprint arXiv:2002.03084*, 2020.

[66] L. Ding, L. Wang, D. Wu, D. Tao, and Z. Tu, "Context-aware cross-attention for non-autoregressive translation," in *COLING*, 2020, pp. 4396–4402.

[67] C. Huang, H. Zhou, O. R. Zaïane, L. Mou, and L. Li, "Non-autoregressive translation with layer-wise prediction and deep supervision," *arXiv preprint arXiv:2110.07515*, 2021.

[68] J. Zhan, Q. Chen, B. Chen, W. Wang, Y. Bai, and Y. Gao, "Non-autoregressive translation with dependency-aware decoder," *arXiv preprint arXiv:2203.16266*, 2022.

[69] C. Shao, J. Zhang, Y. Feng, F. Meng, and J. Zhou, "Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation," in *AAAI*, vol. 34, no. 01, 2020, pp. 198–205.

[70] G. Marjan, V. Karpukhin, L. Zettlemoyer, and O. Levy, "Aligned cross entropy for non-autoregressive machine translation," in *ICML*. PMLR, 2020, pp. 3515–3523.

[71] G. Liu, Z. Yang, T. Tao, X. Liang, Z. Li, B. Zhou, S. Cui, and Z. Hu, "Don't take it literally: An edit-invariant sequence loss for text generation," *arXiv preprint arXiv:2106.15078*, 2021.

[72] Q. Ran, Y. Lin, P. Li, and J. Zhou, "Learning to recover from multi-modality errors for non-autoregressive neural machine translation," in *ACL*, 2020, pp. 3059–3069.

[73] M. Wang, J. Guo, Y. Wang, D. Wei, H. Shang, C. Su, Y. Chen, Y. Li, M. Zhang, S. Tao *et al.*, "Diformer: Directional transformer for neural machine translation," *arXiv preprint arXiv:2112.11632*, 2021.

[74] B. Wei, M. Wang, H. Zhou, J. Lin, and X. Sun, "Imitation learning for non-autoregressive neural machine translation," in *ACL*, 2019, pp. 1304–1312.

[75] Z. Li, Z. Lin, D. He, F. Tian, T. Qin, L. Wang, and T.-Y. Liu, "Hint-based training for non-autoregressive machine translation," in *EMNLP/IJCNLP (1)*, 2019.

[76] L. Tu, R. Y. Pang, S. Wiseman, and K. Gimpel, "Engine: Energy-based inference networks for non-autoregressive machine translation," in *ACL*, 2020, pp. 2819–2826.

[77] Z. Sun and Y. Yang, "An em approach to non-autoregressive conditional sequence generation," in *ICML*. PMLR, 2020, pp. 9249–9258.

[78] Y. Hao, S. He, W. Jiao, Z. Tu, M. Lyu, and X. Wang, "Multi-task learning with shared encoder for non-autoregressive machine translation," in *NAACL-HLT*, 2021, pp. 3989–3996.

[79] J. Liu, Y. Ren, X. Tan, C. Zhang, T. Qin, Z. Zhao, and T.-Y. Liu, "Task-level curriculum learning for non-autoregressive neural machine translation," in *IJCAI*, 2021, pp. 3861–3867.

[80] J. Guo, Z. Zhang, L. Xu, H.-R. Wei, B. Chen, and E. Chen, "Incorporating bert into parallel sequence decoding with adapters," in *NeurIPS*, 2020.

[81] Y. Su, D. Cai, Y. Wang, D. Vandyke, S. Baker, P. Li, and N. Collier, "Non-autoregressive text generation with pre-trained language models," in *EACL*, 2021, pp. 234–243.

[82] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[83] Z. Song, H. Zhou, L. Qian, J. Xu, S. Cheng, M. Wang, and L. Li, "switch-GLAT: Multilingual parallel machine translation via code-switch decoder," in *ICLR*, 2022.

[84] X. Liang, L. Wu, J. Li, Y. Wang, Q. Meng, T. Qin, W. Chen, M. Zhang, T.-Y. Liu *et al.*, "R-drop: regularized dropout for neural networks," *NeurIPS*, vol. 34, 2021.

[85] Anonymous, "Non-autoregressive neural machine translation with consistency regularization optimized variational framework," in *Openreview*, 2022.

[86] ——, "Contrastive conditional masked language model for non-autoregressive neural machine translation," in *Openreview*, 2022.

[87] P. Xie, Z. Cui, X. Chen, X. Hu, J. Cui, and B. Wang, "Infusing sequential information into conditional masked translation model with self-review mechanism," in *COLING*, 2020, pp. 15–25.

[88] J. Kreutzer, G. Foster, and C. Cherry, "Inference strategies for machine translation with conditional masking," *arXiv preprint arXiv:2010.02352*, 2020.

[89] L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer, "Fast decoding in sequence models using discrete latent variables," in *ICML*. PMLR, 2018, pp. 2390–2399.

[90] J. Libovickỳ and J. Helcl, "End-to-end non-autoregressive neural machine translation with connectionist temporal classification," in *EMNLP*, 2018, pp. 3016–3021.

[91] H. Xia, T. Ge, F. Wei, and Z. Sui, "Lossless speedup of autoregressive translation with generalized aggressive decoding," *arXiv preprint arXiv:2203.16487*, 2022.

[92] W. Chan, N. Kitaev, K. Guu, M. Stern, and J. Uszkoreit, "Kermit: Generative insertion-based modeling for sequences," *arXiv preprint arXiv:1906.01604*, 2019.

[93] Z. Kasner, J. Libovickỳ, and J. Helcl, "Improving fluency of non-autoregressive machine translation," *arXiv preprint arXiv:2004.03227*, 2020.

[94] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T.-Y. Liu, "Incorporating bert into neural machine translation," *arXiv preprint arXiv:2002.06823*, 2020.

[95] Z. Yang, B. Hu, A. Han, S. Huang, and Q. Ju, "Csp: Code-switching pre-training for neural machine translation," in *EMNLP*, 2020, pp. 2624–2636.

[96] A. Babu, A. Shrivastava, A. Aghajanyan, A. Aly, A. Fan, and M. Ghazvininejad, "Non-autoregressive semantic parsing for compositional task-oriented dialog," in *NAACL-HLT*, 2021, pp. 2969–2978.

[97] A. Shrivastava, P. Chuang, A. Babu, S. Desai, A. Arora, A. Zotov, and A. Aly, "Span pointer networks for non-autoregressive task-oriented semantic parsing," in *Findings of EMNLP*, 2021, pp. 1873–1886.

[98] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," *NeurIPS*, vol. 32, 2019.

[99] K. Peng, W. Ping, Z. Song, and K. Zhao, "Non-autoregressive neural text-to-speech," in *ICML*. PMLR, 2020, pp. 7586–7598.

[100] K. Yang, W. Lei, D. Liu, W. Qi, and J. Lv, "Pos-constrained parallel decoding for non-autoregressive generation," in *ACL*, 2021, pp. 5990–6000.

[101] T. Jiang, S. Huang, Z. Zhang, D. Wang, F. Zhuang, F. Wei, H. Huang, L. Zhang, and Q. Zhang, "Improving non-autoregressive generation with mixup training," *arXiv preprint arXiv:2110.11115*, 2021.

[102] Y. Higuchi, N. Chen, Y. Fujita, H. Inaguma, T. Komatsu, J. Lee, J. Nozaki, T. Wang, and S. Watanabe, "A comparative study on non-autoregressive modelings for speech-to-text generation," *arXiv preprint arXiv:2110.05249*, 2021.

[103] F.-H. Yu and K.-Y. Chen, "Non-autoregressive transformer-based end-to-end asr using bert," *arXiv preprint arXiv:2104.04805*, 2021.

[104] X. Song, Z. Wu, Y. Huang, C. Weng, D. Su, and H. Meng, "Non-autoregressive transformer asr with ctc-enhanced decoder input," in *ICASSP*. IEEE, 2021, pp. 5894–5898.

[105] W. Qi, Y. Gong, J. Jiao, Y. Yan, W. Chen, D. Liu, K. Tang, H. Li, J. Chen, R. Zhang *et al.*, "Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining," in *ICML*. PMLR, 2021, pp. 8630–8639.

[106] M. Straka, J. Náplava, and J. Straková, "Character transformations for non-autoregressive gec tagging," in *W-NUT*, 2021, pp. 417–422.

[107] P. Li and S. Shi, "Tail-to-tail non-autoregressive sequence prediction for chinese grammatical error correction," in *ACL-IJCNLP*, 2021, pp. 4973–4984.

[108] Q. Han, Y. Meng, F. Wu, and J. Li, "Non-autoregressive neural dialogue generation," *arXiv preprint arXiv:2002.04250*, 2020.

[109] Y. Zou, Z. Liu, X. Hu, and Q. Zhang, "Thinking clearly, talking fast: Concept-guided non-autoregressive generation for open-domain dialogue systems," in *EMNLP*, 2021, pp. 2215–2226.

[110] S. Agrawal and M. Carpuat, "An imitation learning curriculum for text editing with non-autoregressive models," *arXiv preprint arXiv:2203.09486*, 2022.

[111] E. A. Chi, J. Salazar, and K. Kirchhoff, "Align-refine: Non-autoregressive speech recognition via iterative realignment," in *NAACL-HLT*, 2021, pp. 1920–1927.

[112] N. Chen, P. Zelasko, L. Moro-Velázquez, J. Villalba, and N. Dehak, "Align-denoise: Single-pass non-autoregressive speech recognition," *Proc. Interspeech 2021*, pp. 3770–3774, 2021.

[113] N. Chen, S. Watanabe, J. Villalba, P. Żelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters*, vol. 28, pp. 121–125, 2020.

[114] W. Chan, C. Saharia, G. Hinton, M. Norouzi, and N. Jaitly, "Imputer: Sequence modelling via imputation and dynamic programming," in *ICML*. PMLR, 2020, pp. 1403–1413.

[115] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," *arXiv preprint arXiv:2005.08700*, 2020.

[116] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6224–6228.

[117] J. Nozaki and T. Komatsu, "Relaxing the conditional independence assumption of ctc-based asr by conditioning on intermediate predictions," *arXiv preprint arXiv:2104.02724*, 2021.

[118] K. Deng, Z. Yang, S. Watanabe, Y. Higuchi, G. Cheng, and P. Zhang, "Improving non-autoregressive end-to-end speech recognition with pre-trained acoustic and language models," *arXiv preprint arXiv:2201.10103*, 2022.

[119] R. Fan, W. Chu, P. Chang, and J. Xiao, "Cass-nat: Ctc alignment-based single step non-autoregressive transformer for speech recognition," in *ICASSP 2021*. IEEE, 2021, pp. 5889–5893.

[120] Y. Higuchi, H. Inaguma, S. Watanabe, T. Ogawa, and T. Kobayashi, "Improved mask-ctc for non-autoregressive end-to-end asr," in *ICASSP 2021*. IEEE, 2021, pp. 8363–8367.

[121] Y. Bai, J. Yi, J. Tao, Z. Tian, Z. Wen, and S. Zhang, "Fast end-to-end speech recognition via non-autoregressive models and cross-modal knowledge transferring from bert," *IEEE/ACM TASP 2021*, vol. 29, pp. 1897–1911, 2021.

[122] C. H. Puyuan Liu and L. Mou, "Learning non-autoregressive models from search for unsupervised sentence summarization," in *ACL*, 2022.

[123] I. Kulikov, A. Miller, K. Cho, and J. Weston, "Importance of search and evaluation strategies in neural dialogue modeling," in *INLG*, 2019, pp. 76–87.

[124] L. Cheng, W. Jia, and W. Yang, "An effective non-autoregressive model for spoken language understanding," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 241–250.

[125] Y. Ma and Q. Li, "Exploring non-autoregressive text style transfer," in *EMNLP*, 2021, pp. 9267–9278.

[126] F. Huang, Z. Chen, C. H. Wu, Q. Guo, X. Zhu, and M. Huang, "Nast: A non-autoregressive generator with word alignment for unsupervised text style transfer," in *Findings of ACL-IJCNLP 2021*, 2021, pp. 1577–1590.

[127] H. Lu, Z. Wu, X. Wu, X. Li, S. Kang, X. Liu, and H. Meng, "Vaenar-tts: Variational auto-encoder based non-autoregressive text-to-speech synthesis," *arXiv preprint arXiv:2107.03298*, 2021.

[128] R. Shah, K. Pokora, A. Ezzerg, V. Klimkov, G. Huybrechts, B. Putrycz, D. Korzekwa, and T. Merritt, "Non-autoregressive tts with explicit duration modelling for low-resource highly expressive speech," *arXiv preprint arXiv:2106.12896*, 2021.

[129] P. Liu, Y. Cao, S. Liu, N. Hu, G. Li, C. Weng, and D. Su, "Vara-tts: Non-autoregressive text-to-speech synthesis based on very deep vae with residual attention," *arXiv preprint arXiv:2102.06431*, 2021.

[130] Y. Lee, J. Shin, and K. Jung, "Bidirectional variational inference for non-autoregressive text-to-speech," in *ICLR*, 2020.

[131] Y. Ren, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Revisiting over-smoothness in text to speech," *arXiv preprint arXiv:2202.13066*, 2022.

[132] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[133] A. Łańcucki, "Fastpitch: Parallel text-to-speech with pitch prediction," in *ICASSP 2021*. IEEE, 2021, pp. 6588–6592.

[134] H. Guo, H. Lu, X. Wu, and H. Meng, "A multi-scale time-frequency spectrogram discriminator for gan-based non-autoregressive tts," *arXiv preprint arXiv:2203.01080*, 2022.

[135] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," *NeurIPS*, vol. 33, pp. 8067–8077, 2020.

[136] J. Y. Jae-Sung Bae, T.-J. Bak, and Y.-S. Joo, "Hierarchical and multi-scale variational autoencoder for diverse and natural non-autoregressive text-to-speech," *Proc. Interspeech 2022*, 2022.

[137] M. Post, G. Kumar, A. Lopez, D. Karakos, C. Callison-Burch, and S. Khudanpur, "Improved speech-to-text translation with the fisher and callhome spanish-english speech translation corpus," in *Proceedings of the 10th International Workshop on Spoken Language Translation: Papers*, 2013.

[138] S.-P. Chuang, Y.-S. Chuang, C.-C. Chang, and H.-y. Lee, "Investigating the reordering capability in ctc-based non-autoregressive end-to-end speech translation," in *Findings of ACL/IJCNLP*, 2021.

[139] Inaguma, Y. Higuchi, K. Duh, T. Kawahara, and S. Watanabe, "Orthros: Non-autoregressive end-to-end speech translation with dual-decoder," 2021, pp. 7503–7507.

[140] H. Inaguma, Y. Higuchi, K. Duh, T. Kawahara, and S. Watanabe, "Non-autoregressive end-to-end speech translation with parallel autoregressive rescoring," *arXiv preprint arXiv:2109.04411*, 2021.

[141] S. Dalmia, B. Yan, V. Raunak, F. Metze, and S. Watanabe, "Searchable hidden intermediates for end-to-end models of decomposable sequence tasks," in *NAACL-HLT*, 2021.

[142] J. Shi, J. D. Amith, X. Chang, S. Dalmia, B. Yan, and S. Watanabe, "Highland puebla nahuatl speech translation corpus for endangered language documentation," in *Proc. AmericasNLP*, 2021, pp. 53–63.

[143] H. Inaguma, S. Dalmia, B. Yan, and S. Watanabe, "Fast-md: Fast multi-decoder end-to-end speech translation with non-autoregressive hidden intermediates," *arXiv preprint arXiv:2109.12804*, 2021.

[144] S. Agrawal, W. Xu, and M. Carpuat, "A non-autoregressive edit-based approach to controllable text simplification," in *Findings of ACL-IJCNLP*, 2021, pp. 3757–3769.

[145] B. Yu, Y. Wang, T. Liu, H. Zhu, L. Sun, and B. Wang, "Maximal clique based non-autoregressive open information extraction," in *EMNLP*, 2021, pp. 9696–9706.

[146] S. Yu, J. Tack, S. Mo, H. Kim, J. Kim, J.-W. Ha, and J. Shin, "Generating videos with dynamics-aware implicit generative adversarial networks," in *ICLR*, 2021.

[147] T. Hayashi, W.-C. Huang, K. Kobayashi, and T. Toda, "Non-autoregressive sequence-to-sequence voice conversion," in *ICASSP*. IEEE, 2021, pp. 7068–7072.

[148] A. H. Liu, Y.-A. Chung, and J. Glass, "Non-autoregressive predictive coding for learning speech representations from local dependencies," *arXiv preprint arXiv:2011.00406*, 2020.

[149] J. Kasai, N. Pappas, H. Peng, J. Cross, and N. Smith, "Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation," in *ICLR*, 2020.

[150] Anonymous, "Non-autoregressive machine translation: It's not as fast as it seems," in *Openreview*, 2022.

[151] Y. Oka, K. Sudoh, and S. Nakamura, "Using perturbed length-aware positional encoding for non-autoregressive neural machine translation," *arXiv preprint arXiv:2107.13689*, 2021.

[152] M. Ghazvininejad, O. Levy, and L. Zettlemoyer, "Semi-autoregressive training improves mask-predict decoding," *arXiv preprint arXiv:2001.08785*, 2020.

[153] X. Kong, Z. Zhang, and E. Hovy, "Incorporating a local translation mechanism into non-autoregressive translation," in *EMNLP*, 2020, pp. 1067–1073.

[154] C. Shao, Y. Feng, J. Zhang, F. Meng, X. Chen, and J. Zhou, "Retrieving sequential information for non-autoregressive neural machine translation," in *ACL*, 2019, pp. 3013–3024.

[155] L. Qin, F. Wei, T. Xie, X. Xu, W. Che, and T. Liu, "Gl-gin: Fast and accurate non-autoregressive model for joint multiple intent detection and slot filling," 2021.

[156] D. Wu, L. Ding, F. Lu, and J. Xie, "Slotrefine: A fast non-autoregressive model for joint intent detection and slot filling," *arXiv preprint arXiv:2010.02693*, 2020.

[157] H. Le, R. Socher, and S. C. Hoi, "Non-autoregressive dialog state tracking," *ICLR 2021*, 2020.

[158] A. Mohammadshahi and J. Henderson, "Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 120–138, 2021.

[159] T. Komatsu, "Non-autoregressive asr with self-conditioned folded encoders," *arXiv preprint arXiv:2202.08474*, 2022.

[160] S.-P. Chuang, H.-J. Chang, S.-F. Huang, and H.-y. Lee, "Non-autoregressive mandarin-english code-switching speech recognition," *arXiv preprint arXiv:2104.02258*, 2021.

[161] E. G. Ng, C.-C. Chiu, Y. Zhang, and W. Chan, "Pushing the limits of non-autoregressive speech recognition," *arXiv preprint arXiv:2104.03416*, 2021.

[162] Z. Wang, W. Yang, P. Zhou, and W. Chen, "Wnars: Wfst based non-autoregressive streaming end-to-end speech recognition," *arXiv preprint arXiv:2104.03587*, 2021.

[163] R. Fan, W. Chu, P. Chang, J. Xiao, and A. Alwan, "An improved single step non-autoregressive transformer for automatic speech recognition," *arXiv preprint arXiv:2106.09885*, 2021.

[164] C.-F. Zhang, Y. Liu, T.-H. Zhang, S.-L. Chen, F. Chen, and X.-C. Yin, "Non-autoregressive transformer with unified bidirectional decoder for automatic speech recognition," *arXiv preprint arXiv:2109.06684*, 2021.

[165] F. Yu, H. Luo, P. Guo, Y. Liang, Z. Yao, L. Xie, Y. Gao, L. Hou, and S. Zhang, "Boundary and context aware training for cif-based non-autoregressive end-to-end asr," *arXiv preprint arXiv:2104.04702*, 2021.

[166] P. Guo, X. Chang, S. Watanabe, and L. Xie, "Multi-speaker asr combining non-autoregressive conformer ctc and conditional speaker chain," *arXiv preprint arXiv:2106.08595*, 2021.

[167] T. Wang, Y. Fujita, X. Chang, and S. Watanabe, "Streaming end-to-end asr based on blockwise non-autoregressive models," *arXiv preprint arXiv:2107.09428*, 2021.

[168] N. Chen, S. Watanabe, J. Villalba, P. Zelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters*, vol. 28, pp. 121–125, 2021.

[169] Y. Fujita, S. Watanabe, M. Omachi, and X. Chan, "Insertion-based modeling for end-to-end automatic speech recognition," *arXiv preprint arXiv:2005.13211*, 2020.

[170] Y. Bai, J. Yi, J. Tao, Z. Tian, Z. Wen, and S. Zhang, "Listen attentively, and spell once: Whole sentence generation via a non-autoregressive architecture for low-latency speech recognition," *arXiv preprint arXiv:2005.04862*, 2020.

[171] Z. Tian, J. Yi, J. Tao, Y. Bai, S. Zhang, and Z. Wen, "Spike-triggered non-autoregressive transformer for end-to-end speech recognition," *arXiv preprint arXiv:2005.07903*, 2020.

[172] Y. Nakano, T. Saeki, S. Takamichi, K. Sudoh, and H. Saruwatari, "vtts: visual-text to speech," *arXiv preprint arXiv:2203.14725*, 2022.

[173] Y.-C. Wu, T. Hayashi, T. Okamoto, H. Kawai, and T. Toda, "Quasi-periodic parallel wavegan vocoder: A non-autoregressive pitch-dependent dilated convolution model for parametric speech generation," *Proc. Interspeech 2020*, pp. 3535–3539, 2020.

[174] S. Beliaev and B. Ginsburg, "Talknet 2: Non-autoregressive depth-wise separable convolutional model for speech synthesis with explicit pitch and duration prediction," *arXiv preprint arXiv:2104.08189*, 2021.

[175] C.-M. Chien and H.-y. Lee, "Hierarchical prosody modeling for non-autoregressive speech synthesis," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 446–453.

[176] K. Akuzawa, K. Onishi, K. Takiguchi, K. Mametani, and K. Mori, "Conditional deep hierarchical variational autoencoder for voice conversion," *arXiv preprint arXiv:2112.02796*, 2021.

APPENDIX

EXAMPLE

We show the detailed decoding process of different decoding strategies with a specific example in Figure 11, including auto-regressive decoding, semi-autoregressive decoding, fully non-autoregressive decoding, masked-based iterative decoding, and insertion and deletion-based decoding. Their differences can also be observed clearly.

PERFORMANCE COMPARISON

In the paper, we mainly present the NAT works with their performances on the WMT14 English→German translation task. Here we give a broader comparison of WMT14 English↔German, WMT16 English↔Romanian (EN↔RO), and IWSLT14/16 English↔German translation benchmark datasets. The decoding iterations and the speedup ratio compared to AT models are also reported in Table 2.

RESOURCES

We collect valuable resources for NAT models with their open-source information, including the paper URL, code address (Github), and deep learning tools. Table 3 and Table 4 are the summarized information for the resources of NAT task and other extensive tasks.

Src: Wir sind stolz auf wusere Leistung , aber wir wollen jedes Spiel game .
Tgt: We are proud of our performance , but we want to win every game .
Output of different decoding strategies:
Autoregressive decoding:
Iter.1: We are proud of our performance , but we want to win every game .
Iter.2: We are proud of our performance , but we want to win every game .
Iter.3: We are proud of our performance , but we want to win every game .
...
Iter.15: We are proud of our performance , but we want to win every game .
Semi-autoregressive decoding (k=2):
Iter.1: We are proud of our performance , but we want to win every game .
Iter.2: We are proud of our performance , but we want to win every game .
Iter.3: We are proud of our performance , but we want to win every game .
...
Iter.8: We are proud of our performance , but we want to win every game .
Fully non-autoregressive decoding:
Iter.1: We are proud of our performance , but we want to win every game .
Masked-based iterative decoding(Iteration=4):
Iter.0: We are of of our perform , and and want to to win win games .
Iter.1: We are of of our perform , but and we want to win win game .
Iter.2: We are of of our performance , but we want want win every game .
Iter.3: We are of of our performance , but we want win every game .
Iter.4: We are proud of our performance , but we want to win every game .
Insertion and deletion-based decoding:
Iter.0: We are proud of our performance , but we want to win every game .
Iter.1: We are proud of our performance , but we want to to win every game .
Iter.2: We are are proud of our performance , but we want to win every game
Iter.3: We are proud of our performance , but we want to win every game .
Iter.4: We are proud of our performance , but we want to win every game .
Iter.5: We are proud of our performance , but we want to win every game .

Fig. 11. Cases of several different decoding strategies discussed in this paper. Texts marked in yellow denote the content that will be masked and generated in next iteration.

TABLE 2
Performances on popular datasets, i.e., WMT'16 EN↔RO, WMT'14 EN↔DE, and IWSLT'14/16 EN↔DE. "*" indicates training with sequence-level knowledge distillation from a big Transformer; "†" denotes training without sequence-level knowledge distillation; "‡" refers to results on IWSLT'16.

Model	Iteration	Speedup	WMT'14		WMT'16		IWSLT'14/16	
			EN→DE	DE→EN	EN→RO	RO→EN	EN→DE	DE→EN
FT-NAT [16]	1	15.6x	17.69	21.47	27.29	29.06	26.52‡	-
RefineNAT [29]	10	1.5x	21.61	25.48	29.32	30.19	27.11‡	32.31‡
RDP [20]	2.5	3.5x	27.8*	-	-	33.8	-	-
LRF [48]	2.5	3.5x	28.2*	-	-	33.8	-	-
SDMRT [21]	10	-	27.72*	31.65*	33.72	33.94	27.49	-
MD [49]	1	-	25.73	30.18	31.96	33.57	-	-
perLDPE [151]	Adaptive	-	26.3	29.5	-	-	-	-
Glat [37]	1	15.3x	25.21	29.84	31.19	32.04	-	29.61†
PMG [51]	2.5	3.5x	27.8*	-	-	33.8*	-	-
latent-Glat [50]	1	11.3x	26.64	29.93	-	-	-	32.47
Insertion Transformer [53]	$\approx log_2(N)$	-	27.41	-	-	-	-	-
Levenshtein [54]	Adaptive	4.0x	27.27	-	-	33.26	-	-
CMLM [18]	10	1.7x	27.03*	30.53*	33.08	33.31	-	-
SMART [152]	10	1.7x	27.65*	31.27*	-	-	-	-
Disco [34]	Adaptive	3.5x	27.34*	31.31*	33.22	33.25	-	-
JM-NAT [55]	10	5.7x	27.69*	32.24 *	33.52	33.72	-	32.59
Transformer(12-1) [149]	N	2.5x	28.3*	31.8*	33.8	34.8	-	-
MvSR-NAT [52]	10	3.8x	27.39*	31.18*	33.38	33.56	-	32.55
Rewrite-NAT [26]	2.3	3.9x	27.83*	31.52*	33.63	34.09	-	-
CMLMC [56]	10	-	28.37*	31.41*	34.57	34.13	28.51	34.78
FlowSeq [57]	1	1.1x	23.72	28.39	29.73	30.72	27.55	-
NART-DCRF [25]	1	10.4x	23.44	27.22	-	-	-	27.44
PNAT [58]	1	7.3x	23.05	27.18	-	-	-	31.23‡
SynST [59]	$N/6$	4.6x	20.74	25.50	-	-	23.82	-
LaNAT [60]	1	6.8x	25.10	-	-	-	-	-
Imputer [36]	8	3.9x	28.2*	31.8*	34.4	34.1	-	-
LAT [153]	4	6.7x	27.35	32.04	32.87	33.26	-	34.08
AligNART [62]	1	13.2x	26.4	30.4	32.5	33.1	-	-
Reorder-NAT [61]	1	6.0×	22.79	27.28	29.30	29.50	25.29‡	-
CNAT [63]	1	10.4x	25.56*	29.36*	-	-	-	31.15
SNAT [64]	1	22.6x	24.64*	28.42*	32.87	32.21	-	-
Fully-NAT [38]	1	16.5x	27.49	31.39	33.79	34.16	-	-
ENAT [23]	1	25.3x	20.65	23.02	30.08	-	-	24.13
NAT-REG [22]	1	27.6x	20.65	24.77	-	-	23.14‡	23.89
LAVA NAT [65]	1	20.2x	27.94	31.33	-	32.85	-	33.59†
CCAN [66]	10	-	27.5*	-	-	33.7	-	-
DSLP [67]	1	14.8x	27.02	31.61	34.17	34.60	-	-
DAD [68]	1	14.7×	27.51	31.96	34.68	34.98	-	-
CTC [36]	1	18.6×	25.7	28.10	32.20	31.60	-	-
RSI-NAT [154]	1	3.6x	22.27	27.25	30.57	30.83	27.78‡	-
BoN-Joint [69]	1	9.6x	20.90	24.61	28.31	29.29	25.72‡	-
AXE-NAT [70]	1	15.3x	23.53*	27.90*	30.75	31.54	-	-
OAXE-NAT [39]	1	15.3x	26.10*	30.20*	32.40	33.30	-	-
Semi-NAT [17]	$N/2$	1.5x	26.90	-	-	-	-	-
RecoverSAT [72]	$N/2$	2.1x	27.11	31.67	32.92	33.19	30.78‡	-
Unify [35]	10	-	26.24	-	-	-	-	30.73
GAD [91]	1.6	14.3×	26.48	-	-	-	-	-
Diformer [73]	10	-	27.99	31.68	34.37	33.34	-	-
Imitate-NAT [74]	1	18.4x	22.44*	25.67*	28.61	28.90	28.41‡	-
Hint-NART [75]	1	30.2x	21.11	25.24	-	-	-	25.55
ENGINE [76]	10	-	-	-	-	34.04	-	33.17
EM+ODD [77]	1	16.4x	24.54	27.93	-	-	-	30.69
FCL-NAT [24]	1	28.9x	21.70	25.32	-	-	-	26.62
MULTI-TASK NAT [78]	10	-	27.98*	31.27*	33.80	33.60	-	-
TCT-NAT [79]	1	27.6x	21.94	25.62	-	-	26.01‡	28.16
AB-Net [80]	-	2.4x	28.69*	33.57*	-	35.63	-	36.49
Bert+CRF-NAT [81]	1	11.3x	-	-	-	-	-	30.45
CeMAT [40]	10	-	27.2	29.9	33.3†	33.0†	26.7†	33.7†

TABLE 3
A collection of NAT published papers and codes.

Method	Paper URL	Code URL	Framework
Machine Translation			
FT-NAT [16]	https://arxiv.org/pdf/1711.02281.pdf	https://github.com/salesforce/nonauto-nmt	Pytorch
RefineNAT [29]	https://aclanthology.org/D18-1149.pdf	https://github.com/nyu-dl/dl4mt-nonauto	Pytorch
RDP [20]	https://arxiv.org/pdf/2012.14583v2.pdf	-	-
LRF [48]	https://arxiv.org/pdf/2106.00903.pdf	https://github.com/longyuewangdcu/RLFW-NAT	To be released
SDMRT [21]	https://arxiv.org/pdf/2112.11640v1.pdf	-	-
MD [49]	https://aclanthology.org/2020.acl-main.171.pdf	-	-
Glat [37]	https://aclanthology.org/2021.acl-long.155.pdf	https://github.com/FLC777/GLAT	Pytorch/Fairseq
PMG [51]	https://aclanthology.org/2021.findings-acl.247.pdf	-	-
latent-Glat [50]	https://arxiv.org/pdf/2204.02030.pdf	https://github.com/baoy-nlp/Latent-GLAT	Pytorch
Insertion Transformer [53]	https://arxiv.org/pdf/1902.03249.pdf	https://github.com/pytorch/fairseq	Pytorch/Fairseq
Levenshtein [54]	https://arxiv.org/pdf/1905.11006v1.pdf	https://github.com/pytorch/fairseq	Pytorch/Fairseq
CMLM [18]	https://aclanthology.org/D19-1633.pdf	https://github.com/facebookresearch/Mask-Predict	Pytorch/Fairseq
SMART [152]	https://arxiv.org/pdf/2001.08785.pdf	-	-
Disco [34]	https://arxiv.org/pdf/2001.05136.pdf	https://github.com/facebookresearch/DisCo	Pytorch/Fairseq
JM-NAT [55]	https://aclanthology.org/2020.acl-main.36.pdf	https://github.com/lemmonation/jm-nat	Pytorch/Fairseq
Transformer(12-1) [149]	https://arxiv.org/pdf/2006.10369.pdf	https://github.com/jungokasai/deep-shallow	Pytorch/Fairseq
MvSR-NAT [52]	https://arxiv.org/pdf/2108.08447.pdf	-	-
Rewrite-NAT [26]	https://aclanthology.org/2021.emnlp-main.265.pdf	https://github.com/xwgeng/RewriteNAT	Pytorch/Fairseq
CMLMC [56]	https://openreview.net/pdf?id=I2Hw58KHp8O	-	-
FlowSeq [57]	https://arxiv.org/pdf/1909.02480v1.pdf	https://github.com/XuezheMax/flowseq	Pytorch
NART-DCRF [25]	https://arxiv.org/pdf/1910.11555.pdf	-	-
PNAT [58]	https://arxiv.org/pdf/1911.10677.pdf	-	-
SynST [59]	https://aclanthology.org/P19-1122.pdf	https://github.com/dojoteef/synst	Pytorch
LaNAT [60]	https://arxiv.org/pdf/1908.07181v1.pdf	- https://github.com/zomux/lanmt	Pytorch
Imputer [36]	https://aclanthology.org/2020.emnlp-main.83.pdf	https://github.com/rosinality/imputer-pytorch	Pytorch
NAT-with-Local-AT [94]	https://arxiv.org/pdf/2011.06132.pdf	https://github.com/shawnkx/NAT-with-Local-AT	Pytorch
AligNART [62]	https://aclanthology.org/2021.emnlp-main.1.pdf	-	-
Reorder-NAT [61]	https://arxiv.org/pdf/1911.02215.pdf	https://github.com/ranqiu92/ReorderNAT	Pytorch/OpenNMT
CNAT [63]	https://aclanthology.org/2021.naacl-main.458.pdf	https://github.com/baoy-nlp/CNAT	Pytorch
SNAT [64]	https://aclanthology.org/2021.eacl-main.105.pdf	-	-
Fully-NAT [38]	https://aclanthology.org/2021.findings-acl.11.pdf	https://github.com/pytorch/fairseq	Pytorch/Fairseq
ENAT [23]	https://arxiv.org/pdf/1812.09664.pdf	-	-
NAT-REG [22]	https://arxiv.org/pdf/1902.10245.pdf	-	-
LAVA NAT [65]	https://arxiv.org/pdf/2002.03084v1.pdf	-	-
CCAN [66]	https://aclanthology.org/2020.coling-main.389.pdf	-	-
DSLP [67]	https://arxiv.org/pdf/2110.07515.pdf	https://github.com/chenyangh/DSLP	Pytorch/Fairseq
DAD [68]	https://arxiv.org/pdf/2203.16266.pdf	https://github.com/zja-nlp/NAT_with_DAD	Pytorch/Fairseq
CTC [27]	https://www.cs.toronto.edu/~graves/icml_2006.pdf	https://github.com/parlance/ctcdecode	C++
RSI-NAT [154]	https://aclanthology.org/P19-1288.pdf	https://github.com/ictnlp/RSI-NAT	Pytorch
BoN-Joint [69]	https://arxiv.org/pdf/1911.09320.pdf	https://github.com/ictnlp/BoN-NAT	Fairseq
AXE-NAT [70]	https://arxiv.org/pdf/2004.01655.pdf	https://github.com/m3yrin/aligned-cross-entropy	Pytorch
EISL [71]	https://arxiv.org/pdf/2106.15078.pdf	https://github.com/guangyliu/EISL	Pytorch/Fairseq
OAXE-NAT [39]	https://arxiv.org/pdf/2106.05093.pdf	https://github.com/tencent-ailab/ICML21_OAXE	Pytorch/Fairseq
Semi-NAT [17]	https://aclanthology.org/D18-1044.pdf	-	-
RecoverSAT [72]	https://aclanthology.org/2020.acl-main.277.pdf	https://github.com/ranqiu92/RecoverSAT	Pytorch/OpenNMT
GAD [91]	https://arxiv.org/pdf/2203.16487v2.pdf	https://github.com/hemingkx	Pytorch/Fairseq
Unify [35]	https://aclanthology.org/2020.coling-main.25.pdf	-	-
Diformer [73]	https://arxiv.org/pdf/2112.11632v2.pdf	-	-
Imitate-NAT [74]	https://aclanthology.org/P19-1125.pdf	-	-
Hint-NART [75]	https://aclanthology.org/D19-1573.pdf	https://github.com/zhuohan123/hint-nart	Pytorch
ENGINE [76]	https://aclanthology.org/2020.acl-main.251.pdf	https://github.com/lifu-tu/ENGINE	Pytorch/Fairseq
EM+ODD [77]	https://arxiv.org/pdf/2006.16378.pdf	https://github.com/Edward-Sun/NAT-EM	Pytorch
FCL-NAT [24]	http://staff.ustc.edu.cn/~linlixu/papers/aaai20a.pdf	https://github.com/lemmonation/fcl-nat	Tensorflow/Tensortotensor
MULTI-TASK NAT [78]	https://aclanthology.org/2021.naacl-main.313.pdf	https://github.com/yongchanghao/multi-task-nat	Pytorch/Fairseq
TCT-NAT [79]	https://www.ijcai.org/Proceedings/2020/0534.pdf	-	-
AB-Net [80]	https://arxiv.org/pdf/2010.06138v1.pdf	https://github.com/lemmonation/abnet	Pytorch/Fairseq
Bert+CRF-NAT [81]	https://aclanthology.org/2021.eacl-main.18.pdf	https://github.com/yxuansu/NAG-BERT	Pytorch/Fairseq
CeMAT [40]	https://arxiv.org/pdf/2203.09210v1.pdf	https://github.com/huawei-noah	To be release

TABLE 4
A collection of other NAR related published papers and codes. GEC denotes grammatical error correction task, TS denotes text simplification task, IE denotes information extraction task, VG denotes video generation task, VC denotes voice conversion task, and SR denotes speech representation task, TST denotes text style transfer task.

Method	Paper URL	Code URL	Framework
General-Purpose Text Generation			
POSPD [100]	https://aclanthology.org/2021.acl-long.467.pdf	https://github.com/yangkexin/pospd	Pytorch/Fairseq
MIST [101]	https://arxiv.org/pdf/2110.11115v1.pdf	https://github.com/kongds/mist	Pytorch/Fairseq
BANG [105]	https://arxiv.org/pdf/2012.15525v3.pdf	https://github.com/microsoft/BANG	
Task-Specific NAR Text Generation			
CG-nAR [109](Dialogue)	https://arxiv.org/pdf/2109.04084v1.pdf	https://github.com/rowitzou/cg-nar	Pytorch/Transformers
NonAR+MMI [108](Dialogue)	https://arxiv.org/pdf/2002.04250v2.pdf	-	-
GL-GIN [155](Dialogue)	https://arxiv.org/pdf/2106.01925v1.pdf	https://github.com/yizhen20133868/GL-GIN	Pytorch
SlotRefine [156](Dialogue)	https://arxiv.org/pdf/2010.02693v2.pdf	https://github.com/moore3930/SlotRefine	Tensorflow
NADST [157](Dialogue)	https://arxiv.org/pdf/2002.08024v1.pdf	https://github.com/henryhungle/NADST	PyTorch
LR-Transformer [124](Dialogue)	https://arxiv.org/pdf/2108.07005v1.pdf	-	-
TtT [107](GEC)	https://arxiv.org/pdf/2106.01609v3.pdf	https://github.com/lipiji/TtT	Pytorch
BERT-GEC [106](GEC)	https://arxiv.org/pdf/2111.09280v1.pdf	https://github.com/ufal	Pytorch
NAST [126](TST).	https://arxiv.org/pdf/2106.02210v1.pdf	https://github.com/thu-coai/NAST	-
KD+CL+ID [125](TST)	https://aclanthology.org/2021.emnlp-main.730.pdf	https://github.com/sunlight-ym/nar_style_transfer	-
Semantic Parsing			
Span Pointer [97]	https://arxiv.org/pdf/2104.07275v3.pdf	-	-
LightConv [96]	https://arxiv.org/pdf/2104.04923v1.pdf	https://github.com/facebookresearch/pytext	Pytorch/Pytest
RNGTr [158]	https://arxiv.org/pdf/2003.13118v2.pdf	https://github.com/idiap/g2g-transformer	Pytorch
Automatic Speech Recognition			
CTC/attention [118]	https://arxiv.org/pdf/2201.10103v2.pdf	-	-
S-CFE CTC [159]	https://arxiv.org/pdf/2202.08474v1.pdf	-	-
CASSNAT [119]	https://arxiv.org/pdf/2010.14725v2.pdf	-	-
DLP [120]	https://arxiv.org/pdf/2010.13270.pdf	-	-
CTC-enhanced [104]	https://arxiv.org/pdf/2010.15025	-	-
Align-Refine [111]	https://aclanthology.org/2021.naacl-main.154.pdf	https://github.com/amazon-research/align-refine	To be released
Align-Denoise [112]	http://dx.doi.org/10.21437/Interspeech.2021-1906	https://github.com/bobchennan/espnet/tree	Pytorch/Espnet
LASO-BERT [121]	https://arxiv.org/pdf/2102.07594	-	-
P2M [160]	https://arxiv.org/pdf/2104.02258	-	-
Pre-train Comformer [161]	https://arxiv.org/pdf/2104.03416v4.pdf	-	-
WNARS [162]	https://arxiv.org/pdf/2104.03587v2.pdf	-	-
Improved CASS-NAT [163]	https://arxiv.org/pdf/2106.09885v2.pdf	-	-
NAT-UBD [164]	https://arxiv.org/pdf/2109.06684v1.pdf	-	-
Conformer-CIF [165]	https://arxiv.org/pdf/2104.04702	-	-
NAR-BERT-ASR [103]	https://arxiv.org/pdf/2104.04805v1.pdf	-	-
Conditional-Multispk [166]	https://arxiv.org/pdf/2106.08595v1.pdf	https://github.com/pengchengguo/espnet	Pytorch/Espnet
Streaming NAR [167]	https://arxiv.org/pdf/2107.09428v1.pdf	https://github.com/espnet/espnet	Pytorch/Espnet
A-FMLM [168]	https://arxiv.org/pdf/1911.04908.pdf	-	-
Mask-CTC [115]	https://arxiv.org/pdf/2005.08700.pdf	https://github.com/espnet/espnet	Pytorch/Espnet
KERMIT [169]	https://arxiv.org/pdf/2005.13211.pdf	https://github.com/espnet/espnet	Pytorch/Espnet
LSCO [170]	https://arxiv.org/pdf/2005.04862v4.pdf	-	-
Spike-Triggered [171]	https://arxiv.org/pdf/2005.07903v1.pdf	-	-
Intermediate CTC [116]	https://arxiv.org/pdf/2102.03216v1.pdf	https://github.com/espnet/espnet	Pytorch/Espnet
Self-Conditioned CTC [117]	https://arxiv.org/pdf/2104.02724.pdf	https://github.com/espnet/espnet	Pytorch/Espnet
Text to Speech			
BVAE-TTS [130]	https://openreview.net/pdf?id=o3iritJHLfO	https://github.com/LEEYOONHYUNG/BVAE-TTS	Pytorch
vTTS [172]	https://arxiv.org/pdf/2203.14725.pdf	-	-
Gan-TTS [134]	https://arxiv.org/pdf/2203.01080.pdf	https://github.com/yanggeng1995/GAN-TTS	Pytorch
VARA-TTS [129]	https://arxiv.org/pdf/2102.06431v1.pdf	https://github.com/vara-tts/VARA-TTS	-
Glow [135]	https://arxiv.org/pdf/2005.11129.pdf	https://github.com/jaywalnut310/glow-tts	Tensorflow/Tensor2tensor
VAENAR-TTS [127]	https://arxiv.org/pdf/2107.03298v1.pdf	https://github.com/thuhcsi/VAENAR-TTS	Pytorch
ParaNet [99]	https://arxiv.org/pdf/1905.08459.pdf	https://github.com/ksw0306/WaveVAE	Pytorch
WaveGAN [173]	https://arxiv.org/pdf/2005.08654v1.pdf	https://github.com/bigpon/QPPWG	PyTorch
FastSpeech [98]	https://arxiv.org/pdf/1905.09263.pdf	https://github.com/coqui-ai/TTS	PyTorch/TTS
TalkNet2 [174]	https://arxiv.org/pdf/2104.08189v3.pdf	https://github.com/rishikksh20/TalkNet2-pytorch	-
FastSpeech2 [175]	https://arxiv.org/pdf/2011.06465v3.pdf	https://github.com/ming024/FastSpeech2	Pytorch
HiMuV-TTS [136]	https://arxiv.org/pdf/2204.04004.pdf	-	-
Speech translation			
NAR-ST [138]	https://arxiv.org/pdf/2105.04840v1.pdf	https://github.com/voidism/NAR-ST	Pytorch/Espnet
Orthros [139]	https://arxiv.org/pdf/2010.13047	-	-
Orthros-CMLM [140]	https://arxiv.org/pdf/2109.04411v1.pdf	-	-
Fast-MD [143]	https://arxiv.org/pdf/2109.12804v1.pdf	-	-
Others			
PMI-based NAR [144](TS)	https://aclanthology.org/2021.findings-acl.330.pdf	-	-
MacroIE [145](IE)	https://aclanthology.org/2021.emnlp-main.764.pdf	-	-
DIGAN [146](VG)	https://arxiv.org/pdf/2202.10571.pdf	-	-
NAR S2S VC [147](VC)	https://arxiv.org/pdf/2104.06793	-	-
CDHVAE [176](VC)	https://arxiv.org/pdf/2112.02796.pdf	-	-
NPC [148](SR)	https://arxiv.org/pdf/2011.00406v1.pdf	https://github.com/Alexander-H-Liu/NPC	Pytorch