Revisiting the Iterative Non-Autoregressive Transformer

Anonymous ACL submission

Abstract

Iterative non-autoregressive (NAR) models share a spirit of mixed autoregressive (AR) and fully NAR models, seeking a balance between generation quality and inference efficiency. These models have recently demonstrated impressive performance in varied generation tasks, surpassing the autoregressive (AR) Transformer. However, they also face several challenges that impede further development. In this work, we target building more efficient and competitive iterative NAR models by conducting systematic studies and analytical experiments. Firstly, we conduct an oracle experiment and introduce two newly proposed metrics to identify the potential problems existing in current refinement processes, and look back on the various iterative NAR models to find the key factors for realizing our purpose. Subsequently, based on the analyses of the limitations of previous inference algorithms, we propose a simple yet effective strategy to conduct efficient refinements without performance declines. Experiments on five widely used datasets show that our final models significantly outperform all previous NAR models and AR Transformer, even with fewer decoding steps on two datasets.

1 Introduction

Transformer-based models (Vaswani et al., 2017) have achieved promising performance in various tasks, particularly after the emergence and progress of large language models recently (Touvron et al., 2023a; OpenAI, 2023; Touvron et al., 2023b). However, these models adopt an autoregressive (AR) decoding paradigm where tokens are generated one by one in a strict left-to-right order. Consequently, they suffer from low inference efficiency, which even worsens as model parameters increase (Zhao et al., 2023). Non-autoregressive (NAR) models provide an alternative text generation paradigm (Gu et al., 2018). Unlike AR models, NAR models can predict all the target tokens in parallel, significantly reducing inference latency. However, this parallel decoding paradigm also leads to performance degradation due to independent predictions lacking target side dependency (Qian et al., 2021; Xiao et al., 2022; Huang et al., 2023).

Researchers have proposed iterative NAR models to balance generation quality and inference efficiency (Lee et al., 2018; Ghazvininejad et al., 2019; Chan et al., 2020). These models utilize multiple decoding steps to generate the final results and retain the non-autoregressive decoding paradigm in each step. A partial target sequence is proposed in each decoding step and then refined in the subsequent steps. The performance of competitive iterative NAR models achieves significant improvements through iterative refinements, surpassing their AR counterparts (Huang et al., 2022b; Xiao et al., 2023). However, these models have also revealed some flaws in the corresponding research, including failure under specific model structure (Kasai et al., 2020b), declines in inference speedup (Helcl et al., 2022) and the anisotropic problem (Guo et al., 2023), hindering the further development of iterative NAR models.

Therefore, ***how to build more efficient and competitive iterative NAR models*** deserves further exploration. In this paper, we aim to address this question by conducting systematic studies and analytical experiments:

- We conduct in-depth explorations of current iterative NAR models (§3). Specifically, we verify and further quantitatively analyze the potential problems existing in current refinement processes through an oracle experiment (§3.1) and two newly proposed metrics (§3.2). Besides, we conduct analytical experiments based on various iterative NAR models and discover that different enhanced methods play different roles in building efficient and competitive models (§3.3). Then, we attempt to realize our purpose by combining previous superior methods, but notice performance declines with previous efficient strategies (§3.4).

- We trial better strategies for iterative NAR models to become efficient while maintaining competitive performances (§4). We first analyze the limitations of current refinement strategies (§4.1) and then propose a simple yet effective inference algorithm for iterative NAR models (§4.2). Combining it with previous competitive strategies can achieve superior performance with fewer decoding steps.

Experiments on 5 widely used datasets demonstrate the effectiveness of our models. We yield significant performance improvements (around 0.8 BLEU score on average) over the previous best iterative NAR models and realize completely surpassing AR Transformer (over 1 BLEU score on average). Besides, our models only need 4 decoding steps to set new SOTA performance on WMT'14 DE→EN and WMT'16 EN→RO datasets compared with previous ones with 10 decoding steps.

2 Preliminaries

Non-autoregressive Language Model Up to now, most generative models are autoregressive (AR) models which generate the target sequence one by one from a left-to-right order during inference. They adopt AR factorization during training to maximize the following likelihood: $\mathcal{L}_{\text{AR}} = \sum_{t=1}^{T} \log P(y_t|y_{<t}, X; \theta)$, where $y_{<t}$ denotes the previous generated target tokens, T denotes the target length, X is the source sentence, and θ denotes the model parameters. Unlike these AR models, non-autoregressive (NAR) language models generate the target sequence in parallel during inference, which can be further divided into fully NAR models and iterative NAR models according to their decoding steps. Fully NAR models only adopt one step to generate the target sequence, and adopt fully conditional independent factorization during training to maximize the following likelihood: $\mathcal{L}_{\text{F-NAR}} = \sum_{t=1}^{T} \log P(y_t|X; \theta)$. Iterative NAR models adopt multiple decoding steps to generate the target sequence and keep the NAR property in each decoding step. They aim to maximize the following likelihood during training: $\mathcal{L}_{\text{I-NAR}} = \sum_{t \in Y_{tgt}} \log P(y_t|\hat{Y}, X; \theta)$, where Y_{tgt} denotes the prediction target tokens of the current decoding step and the $\hat{Y}$ denotes the generation result of the previous decoding step. Iterative NAR models give the chance to refine the generated result, thus significantly improving the performance compared to fully NAR models.

Conditional Masked Language Model Conditional Masked Language Model (CMLM) is a typical and widely-used iterative NAR model (Ghazvininejad et al., 2019), which adopts a Transformer-based encoder-decoder architecture with some specific modifications in the decoder blocks to support NAR generation manner. During training, CMLM uses masked language modeling tasks like BERT for training. Specifically, given a training pair (X, Y), CMLM first selects partial tokens in Y to be masked, denoted as Y_{mask}, while the unmasked tokens as Y_{obs}. CMLM learns to predict the masked tokens Y_{mask}, and to maximize: $\mathcal{L}_{\text{CMLM}} = \sum_{y_t \in Y_{mask}} \log P(y_t|Y_{obs}, X; \theta)$, where θ denotes the trainable parameters. Besides, CMLM also adopts an auxiliary task to predict the target length. During inference, CMLM utilizes multiple decoding steps to generate an entire sequence in parallel via a specially designed Mask-Predict algorithm. Given the source sentence X and the total T decoding steps, CMLM first predicts the target length L. Then, it sends the entire masked target sequence (i.e., L `[MASK]` tokens since we have no target tokens in the first iteration) into the decoder and predicts them. After each decoding step, the model will choose a specific number of tokens to mask again with the relatively lowest prediction probability from the target sequence. These newly masked tokens Y_{mask} will be re-predicted in the next step. In an intermediate tth step, the number of the newly masked tokens n can be calculated as $n = \frac{T-t}{T} * L$.

Follow-up Methods of CMLM Based on CMLM, researchers have proposed many follow-up enhanced methods from different perspectives to improve the training and inference process, e.g., using the better masking methods (Guo et al., 2020; Xiao et al., 2023) or enhanced modeling mechanism (Kasai et al., 2020a; Cheng and Zhang, 2022; Chen et al., 2024) to replace the traditional uniform masking training strategy, utilizing an additional AR decoder to enhance the NAR modeling during training (Hao et al., 2021; Liang et al., 2022), adopting the Locater module to determine the newly masked tokens during inference (Geng et al., 2021), introducing a self-correction mechanism to enhance the traditional Mask-Predict algorithm (Ghazvininejad et al., 2020; Huang et al., 2022b), and etc. We include more details about these variants in the Appendix A due to the length limitation. In this work, we conduct a comprehensive analysis of the traditional CMLM and these follow-up methods, targeting building more efficient and competitive iterative NAR models.

3 In-depth Explorations of Current Iterative NAR Models

In this section, we conduct in-depth explorations of current iterative NAR models. Specifically, we introduce several sub-problems and make detailed analyses. We aim to find the key factors for building efficient and competitive iterative NAR models.

Problems and Explorations. Firstly, previous works always focus on the final generated output after pre-defined fixed decoding steps to evaluate iterative NAR models, but overlook the fine-grained analysis of intermediate decoding steps throughout the refinement process. Consequently, some potential problems (e.g., useless and negative decoding steps) during the refinement process can not be reflected based on the current evaluation process. Naturally, we wonder: ***how well do current refinement strategies perform for iterative NAR models*** (§3.1). We compare the performance achieved with the current refinement algorithm and that under an ideal setting. Furthermore, to quantitatively analyze the potential problems mentioned above, we introduce two metrics (DRR and ROR) to evaluate the stability of each decoding step and the reliability of the whole refinement process. We aim to answer ***how to better evaluate the refinement process of different iterative NAR models*** (§3.2). Based on our proposed two metrics, we compare different iterative NAR models under a consistent re-implementation. We aim to find ***what are the key components for iterative NAR models to perform better*** (§3.3). Finally, we conduct extended experiments to answer ***can better performance be achieved by combining superior methods*** (§3.4), and make a summary (§3.5).

Experimental Settings. We adopt the vanilla CMLM and several typical variants which contain different improving strategies from different respects as mentioned in Section 2 for exploration. We summarize them as different categories: adopting enhanced training skills (JM-NAT, AMOM, Multitask-NAT), using adaptive inference algorithms (Disco, Rewrite-NAT), and introducing self-correction mechanism (SMART, CORR, CMLMC). To make more consistent comparisons, we re-implement all these models based on the same hardware and training hyper-parameters. For the evaluation dataset, we select the IWSLT'14 DE$\rightarrow$EN dataset containing about $170k$ training sentence pairs, $7k$ valid pairs, and $7k$ test pairs. We train each model on the training set and then evaluate them on the test set. Following the previous work (Kasai et al., 2020a), we apply sequence-level knowledge distillation (Kim and Rush, 2016) for all backbone models. All experiments use the Fairseq library (Ott et al., 2019) with GTX 3090 GPU cards. We adopt the same training hyper-parameters following CMLM realization in Fairseq. During inference, we average the 5 best checkpoints chosen by validation BLEU as our final model. Finally, we evaluate the generation quality with BLEU score (Papineni et al., 2002). Besides, to eliminate the effects of randomness, we follow the previous works to use statistical significance tests (Koehn, 2004) to detect if the difference in BLEU score between the traditional CMLM and other enhanced iterative NAR models is significant.

3.1 *How Well do Current Refinement Strategies Perform for Iterative NAR Models?*

Exploration Process. Firstly, we design an oracle experiment with ideal settings in which we can select the best-generated output from different decoding steps for each testing instance. Specifically, we adopt 10 decoding steps during inference following the common practice. Then, rather than adopting the generated output of the last decoding step for each test instance, we select the one with the highest evaluation score (e.g., sentence BLEU) as the final generated result. This setting can eliminate the impacts of the above-mentioned potential problems (e.g., useless and negative decoding steps) in the refinement process. Finally, we compare the results of this oracle experiment with those achieved from the original settings. In this experiment, we adopt two current main-stream models with different refinement strategies: the CMLM with the Mask-Predict algorithm and CORR with the self-correction algorithm.

Main Findings. Results of the oracle experiment and with original settings are shown in Table 1 (Choose Best v.s. Original). We can find: (1) There exists much space for the improvements of current refinement methods, and the performance through best choice outperforms that from the last decoding step over 2.5 BLEU score. (2) More superior results appear during the CORR refinement process, which indicates that the self-correction algorithm can bring some benefits. Besides, we should recognize that we can not realize the ideal settings of the above oracle experiment since there is no ground truth during our inference process. However, we can still verify that some problems exist in the current refinement process. It also motivates us to explore better refinement strategies in which we can effectively reduce or even avoid useless and negative refinement decoding steps.

Model	Original	Choose Best
CMLM	33.55	36.14
CORR	33.76	36.45

Table 1: BLEU score of the oracle experiment (Choose Best) and with original settings (Original).

3.2 *How to Better Evaluate the Refinement Process of Different Iterative NAR Models?*

As the above exploration shows, the potential problems in the refinement process (e.g., useless and negative decoding steps) are serious in the current iterative NAR models. However, the current evaluation process, where we adopt the generated output of the last decoding step, can not directly reflect these potential problems. Therefore, we introduce two metrics, Decline Risks of Refinements (DRR) and Ratio of Over-Refinements (ROR), to respectively measure the extent of these potential problems and evaluate the stability and reliability of the refinement process.

Decline Risks of Refinements. Decline Risks of Refinements (DRR) evaluates the stability of the refinement process of iterative NAR models. It measures the performance decline rate after one specific decoding step, i.e., the extent of the negative decoding step. Specifically, given a test set with N examples, a fixed decoding step T, we compute the ratio of each example during the whole refinement process where the performance declines compared with the previous iteration, formatted as:

$$\mathrm{DRR} = \frac{1}{T-1}\sum_{t=1}^{T-1}\frac{|\textbf{Score}_i^t > \textbf{Score}_i^{t+1}|}{N}, \quad (1)$$

where Score_i^t denotes the performance of sample i in the tth step.

Ratio of Over-Refinements. Ratio of Over-Refinements (ROR) evaluates the reliability of the final generated output in iteration T. It measures the failure rate of the output from the last decoding step to be the best, i.e., the extent of the useless decoding steps. Specifically, given a test set with N examples, a fixed decoding step T, we compute the ratio of each example whose best performance is achieved in the intermediate steps of the refinement process, formatted as:

$$\mathrm{ROR} = \frac{1}{T-1}\sum_{t=1}^{T-1}\frac{|\textbf{Score}_i^t > \textbf{Score}_i^{T}|}{N}, \quad (2)$$

where Score_i^t denotes the performance of sample i in the tth step, Score_i^T denotes the performance of sample i in the final iteration T.

3.3 *What are the Key Components for Iterative NAR Models to Perform Better?*

Exploration Process. We look for the key components for two aspects, i.e., ***efficient*** and ***competitive***. The former can be reflected in the stability and reliability of the refinement process with our proposed metrics, and the latter can be reflected in the final performance. We evaluate the related enhanced CMLM methods based on our re-implementations. For the models with adaptive inference algorithms (Disco and RewriteNAT), in Equation 1 and Equation 2, we set T as the adaptive decoding step of each sentence pair during inference, and 10 for other methods following the previous works. Besides, for the models that support two inference algorithms (e.g., CMLMC can omit the self-correction process and change to the original Mask-Predict algorithm), we both report the results with the Mask-Predict algorithm and the corresponding enhanced inference strategy.

Main Findings. The results are presented in Table 2, we find that: (1) ***DRR and ROR are relatively lower while decoding with adaptive inference algorithms.*** These models aim to find more suitable methods to decide how many and which tokens to mask, and when to stop refinements during inference. They can achieve comparable performance with fewer decoding steps, indicating that adaptive inference algorithms bring benefits to building more efficient iterative NAR models. (2) ***Enhanced***

training skills bring benefits on generation quality, but there is no evident improvement on DRR and ROR. These models trained with enhanced training skills can improve performance compared with the vanilla CMLM, but DRR and ROR are still relatively high, indicating that enhanced training skills are useful for building more competitive iterative NAR models, the performance improvements of these models come from the better ability to model token dependency during training rather than stabilizing the refinement process. (3) ***Introducing the self-correction mechanism can improve performance, but DRR gets higher.*** These models with the self-correction mechanism can achieve around one BLEU score improvement. However, DRR increases, indicating that the self-correction mechanism may bring more unstable factors during the refinement process.

3.4 *Can Combining Superior Methods Bring Benefits?*

Exploration Process. We can learn from the explorations in Section 3.3 that different enhanced methods are independently beneficial to making the models more efficient and competitive. Naturally, we wonder: can combining superior methods bring benefits? We further explore the following questions: (1) Since the adaptive inference algorithms can bring promising performance with fewer decoding steps, can they further improve the performance with more steps? (2) Since adopting enhanced training skills and the self-correction mechanism can boost performance but not stabilize the refinement process, can we incorporate the adaptive inference algorithms into these models to make them more efficient? Specifically, for question 1, we force these models (Disco and RewriteNAT) to continue the refinement process until reaching the maximal T decoding step. For question 2, we first combine the previous superior methods of enhanced training skills and adaptive inference algorithms (AMOM and CMLMC, denoted as AMOMC), and then we further apply the Locator module proposed in RewriteNAT into AMOMC.

Main Findings. The results are shown in Table 2, we can find that: (1) Concerning the models with adaptive inference algorithms, the performance even declines once we adopt more decoding steps for them, e.g., the performance declines from 33.32 to 33.22 for Disco, from 33.91 to 33.88 for RewriteNAT. Besides, DRR and ROR get much higher with more decoding steps, indicating that models with adaptive inference algorithms do not need many decoding steps to achieve the best performance during inference. (2) Further utilizing the Locator module for AMOMC can make the refinement process more efficient since it can achieve comparable performance with fewer decoding steps and get lower DRR and ROR, but it also leads to performance declines compared with the original AMOMC.

Methods	Iteration	BLEU	DRR (%)	ROR (%)
Enhanced Training Skills				
CMLM	10	33.55	13.4	19.1
JM-NAT	10	32.60	14.4	17.5
Multitask-NAT	10	33.60	16.5	18.4
Disco	10	33.22	14.6	13.1
RewriteNAT †	10	33.88	12.1	14.4
CORR †	10	33.65	13.3	14.1
CMLMC †	10	34.02	13.1	13.8
AMOM †	10	**34.68**	16.3	17.9
Adaptive Inference Algorithms				
Disco	Adv.	33.32	11.8	6.9
RewriteNAT †	Adv.	33.91	**7.9**	**1.1**
Self-correction Mechanism				
SMART	10	33.17	14.5	16.6
CORR †	10	33.76	15.0	15.3
CMLMC †	10	**34.40**	15.2	14.9
Combining Superior Methods				
AMOMC †	10	**35.08**	16.8	16.7
w/ Locator †	Adv.	**34.68**	**5.9**	**6.0**

Table 2: DRR and ROR of different models. Adv. denotes adaptive decoding steps, which is always less than 10. † denotes that the BLEU improvements over CMLM are statistically significant with $p < 0.05$.

3.5 Summary

Now, we summarize our above explorations. We first analyze the potential problems existing in current refinement methods through an oracle experiment and two proposed metrics. We encourage the researchers to pay more attention to the intermediate decoding steps. Next, we conduct comparative experiments to look for the key components for building more efficient and competitive iterative NAR models, and then further combine superior methods to realize our purpose. However, we find that the current efficient strategy leads to performance declines. This motivates us to explore better strategies for building efficient iterative NAR models while maintaining competitive performance.

4 Trials for Better Efficient Strategies

In this section, we explore better strategies for iterative NAR models to become efficient in the refinement process while maintaining competitive performance. We conduct a detailed analysis of original refinement methods and then propose a simple yet effective strategy to realize our purpose.

Problems and Explorations. Firstly, the Mask-Predict algorithm exhibits higher DRR and ROR than adaptive inference algorithms in Table 2. Therefore, we aim to explore: ***what makes the Mask-Predict algorithm fail to do efficient refinements*** (§4.1). Besides, although current adaptive inference algorithms are advantageous for reducing the decoding steps, they also lead to performance declines. Therefore, we analyze the corresponding reasons and further investigate: ***are there more effective inference algorithms for iterative NAR models*** (§4.2). Finally, we analyze the aforementioned questions and point out future directions for iterative NAR models (§4.3).

Experimental Settings. During the analysis on the failure of the Mask-Predict algorithm, we adopt the CMLM checkpoint achieved from the above exploration process. For the explorations of more effective inference algorithms, we adopt more datasets except IWSLT'14 DE→EN to evaluate our proposed methods. Specifically, we choose two WMT datasets that are widely used in previous NAR works, WMT'16 English→Roman (En↔Ro) and WMT'14 English→German (En↔De) language pairs. The training data sizes are about 0.6M and 4.5M for En↔Ro and En↔De. The test data are from the corresponding newest data, which contains around 3,000 and 7,000 samples, respectively. Besides, the training and evaluation settings are the same as those mentioned in Section 3.

4.1 *What Makes the Mask-Predict Algorithm Fail to Do Efficient Refinements?*

We attribute the success of the adaptive inference algorithm to the reasonable strategy to determine *"which token should be masked in the next decoding step?"* Comparatively, the Mask-Predict algorithm relies on predicted confidence to select masked tokens in the subsequent decoding step. However, we have identified two shortcomings with this confidence-based refinement process:

1) ***The independent confidence updating strategy for each token is sub-optimal.*** In the Mask-Predict algorithm, the prediction confidence is updated only for masked tokens during each decoding step. On the other hand, the confidence for unmasked tokens remains the same as the last decoding step when it was predicted. This denotes that the prediction confidences of masked and unmasked tokens are derived from different decoding steps and under different masking conditions. Consequently, this inconsistency poses challenges in determining which tokens should be masked in the subsequent decoding step. This shortcoming is also supported by the comparison presented in Table 2. Several models which can update the confidence scores of all the tokens in the same decoding step can alleviate this problem to some extent, e.g., Disco, RewriteNAT, and CMLMC all achieve lower DRR and ROR even without adopting adaptive inference algorithms during inference.

2) ***The prediction confidence of CMLM is not strongly related to the generation quality.*** As discussed in Section 2, CMLM selects the prediction probability as the confidence to choose newly masked tokens. This approach assumes that tokens with higher prediction probability scores are more reliable. However, previous works have also highlighted several issues. Ding et al. observe that some specific tokens, such as high-frequency words and conjunctions, consistently exhibit high confidence, leading to repetitive output and neglect of low-frequency but important words. Additionally, Liang et al. note that the function words dominate the high probability region of the output distribution, making it challenging to generate informative tokens using the Mask-Predict algorithm with CMLM. However, no substantial experiment exists to present the irrelevance between the prediction confidence and final generation output. Thus, we perform a simple experiment to verify this.

Exploration Process. We explore the confidence distribution during inference. We first randomly mask several tokens in the target sequence and send them into CMLM to obtain the prediction confidence. Then, since the Mask-Predict algorithm always selects tokens with the highest prediction probability, we wonder whether the probability of masked ground truth tokens ranks first, e.g., given the test sentence *"Thank you."* We first replace the token *"you"* with the `[MASK]` token, then we send the sequence *"Thank* `[MASK]` *."* into CMLM, and verify whether the prediction probability of token *"you"* ranks first. If not, the highest prediction confidence does not equal the correct token.

Main Findings. We conduct analytic experiments on the validation and test set. Results are

	Model	Iterations	WMT'14		WMT'16	
			EN→DE	DE→EN	EN→RO	RO→EN
AR	Transformer (Vaswani et al., 2017)	*N*	27.30	31.29	-	-
	Transformer*	*N*	28.41	32.28	34.23	34.28
Iterative NAR	Refine-NAT (Lee et al., 2018)	10	21.61	25.48	27.11	30.19
	Levenshtein (Gu et al., 2019)	Adv.	27.73	-	33.02	-
	CMLM (Ghazvininejad et al., 2019)	10	27.03	30.53	33.08	33.31
	DisCo (Kasai et al., 2020a)	Adv.	27.34	-	33.25	33.22
	SMART (Ghazvininejad et al., 2020)	10	27.65	31.27	33.85	33.53
	JM-NAT (Guo et al., 2020)	10	27.69	32.24	33.52	33.72
	RDP (Ding et al., 2020)	10	27.80	-	33.70	-
	LFR (Ding et al., 2021)	10	27.80	-	-	33.90
	RewriteNAR (Geng et al., 2021)	Adv.	27.83	31.52	33.63	34.09
	MvCR-NAT (Xie et al., 2021)	10	27.39	31.18	33.38	33.56
	CORR (Huang et al., 2022b)	10	28.19	31.31	34.31	34.08
	CMLMC (Huang et al., 2022b)	10	28.37	31.41	34.57	34.13
	CCMLM (Cheng and Zhang, 2022)	10	27.93	31.57	33.88	34.18
	AMOM (Xiao et al., 2023)	10	27.57	31.67	34.62	34.82
	EECR (Chen et al., 2024)	10	28.04	31.65	34.33	34.32
*Ours**	AMOMC	4	28.35	32.72	34.80	35.08
		10	28.90	33.25	35.01	35.26
	AMOMC + ARSCORER †	4	28.82	**33.25**	**35.15**	35.15
		10	**29.17**	**33.33**	**35.27**	**35.48**

Table 3: Results on 4 WMT machine translation tasks. * denotes the results of our implementations. † denotes that the BLEU improvements over AMOMC are statistically significant with $p < 0.05$.

Set	Win (%)	Lose (%)
Valid	54.61	45.39
Test	54.10	45.90

Table 4: **Win** denotes the model predicts the ground truth token as the final results, **Lose** denotes the vice.

shown in Table 4. We find that only around 54 percent of tokens meet our expectations, i.e., these ground truth tokens have the highest prediction probabilities. This shows that the prediction confidence achieved from the model itself is not strongly related to the correct tokens. This also provides evidence that utilizing an extra module to score the predicted tokens, such as the Locator, proves to be more effective than the model itself. We attribute this failure to the conditional independent factorization for CMLM learning, which causes CMLM to fail to capture the target-side dependency well during training (Gu and Kong, 2021).

4.2 *Are There More Effective Inference Algorithms for Iterative NAR Models?*

The explorations in Section 4.1 explain that why adaptive inference algorithms are more effective than the traditional Mask-Predict algorithm. However, noticing that adopting the Locater also leads to performance declines, we first analyze the corresponding reason. Since the Locator module assigns zero-one discrete scores for predicted tokens, i.e., the token will be masked again in the next decoding step once it is scored as zero, and not be masked if it is scored as one. We point out that this scoring mechanism is too absolute, e.g., there is no difference for unreliable tokens which are all scored as zero, and once the scores for all tokens are one, there are no subsequent actions for further improving the generation quality. To explore the potential of a more effective extra scoring module for iterative NAR models, we intended to replace the zero-one discrete score with a zero-one continuous distribution, in which we can design the refinement process more flexibly and constantly.

Exploration Process. We aim to find a simple yet effective mechanism to score each token within a sentence, and then we can depend on these scores to determine which tokens should be masked in the subsequent decoding step. Motivated by the previous practice that a pre-trained AR model can successfully serve as an effective scorer on the sentence-lever to evaluate the fluency of sentences, we can extend it as a token-level scorer, named ARSCORER in the remaining space of this paper. Specifically, we utilize the generated tokens from each decoding step as inputs for a pre-trained AR model. The AR model conducts its prediction on this input sequence in an autoregressive manner. Subsequently, we obtain the corresponding prediction distribution and use the probability associated with the input token index as the final score. The scores range from zero to one after undergoing the normalized softmax operation. Comparatively, adopting ARSCORER offers several advantages over the Mask-Predict algorithm, which have also been mentioned in the previous section: (1) The AR model can assess the validity of each token in the whole sentence and update the corresponding prediction probability of each token after each decoding step of NAR model. (2) Previous studies have shown that models trained with autoregressive factorization excel in capturing target side dependencies compared to NAR models (Huang et al., 2022a). Besides, these AR models do not suffer from the multi-modality problem. Therefore, adopting extra ARSCORE to provide the prediction score is more robust and effective.

Main Findings. The results on various WMT datasets are shown in Table 3, we can find that: (1) Combining superior methods (AMOMC) achieves significant performance improvements, outperforming all baseline models around 0.8 BLEU score. (2) Adopting ARSCORER can quickly achieve competitive performance, i.e., it can get comparable even better performance with only 4 decoding steps compared with AMOMC with 10 decoding steps, outperforming all baseline models and AR counterparts significantly. (3) Adopting ARSCORER outperforms AMOMC in all evaluation settings, especially with relatively fewer decoding steps, indicating ARSCORER can bring benefit for building efficient iterative NAR models.

Further Analysis. We further compare the backbones models with those with ARSCORER based on our proposed two metrics, DRR and ROR, as mentioned in Section 3.2. Results on IWSLT'14 DE→EN and WMT'16 EN→RO datasets are presented in Table 5. We can find that: (1) The models with ARSCORER can achieve lower DRR and ROR compared with the corresponding baselines. (2) DRR and ROR are higher on the WMT'16 EN→RO dataset across all models, indicating that this dataset is relatively difficult to learn.

4.3 Summary

In this section, we aim to explore the potential for better efficient strategies. We begin by examining the limitations of the Mask-Predict algorithm in facilitating consistent and efficient refinements. Through thorough analysis and corresponding experimentation, we attribute these limitations to the independent confidence updating strategies and the unrelated prediction confidence to generation output. Consequently, we endeavor to identify a superior strategy to address these issues. Fortunately, by adopting the pre-trained AR models to serve as a scorer, iterative NAR models can conduct steady and effective refinements, thereby achieving superior performance with even fewer decoding steps, and getting closer to the efficient iterative NAR models. It is worth noting that there are other viable options for scoring, such as adopting a pre-trained language model or even current well-known large language models, we leave this as future work.

Methods	**Iteration**	**BLEU**	**DRR (%)**	**ROR (%)**
IWSLT'14 DE→EN				
CMLM	10	33.55	13.4	19.1
+ ARSCORER	10	34.05	10.0	13.4
AMOMC	10	35.08	16.8	16.7
+ ARSCORER	10	35.61	9.8	13.6
WMT'16 EN→RO				
CMLM	10	33.19	17.1	20.1
+ ARSCORER	10	33.55	10.9	14.8
AMOMC	10	35.03	21.4	24.4
+ ARSCORER	10	35.27	15.8	19.0

Table 5: Results of DRR and ROR with ARSCORER.

5 Conclusion and Future Outlook

In this paper, we conduct extensive experiments and detailed analysis to address: ***how to build more effective and competitive iterative NAR models***. By combining competitive strategies and the newly proposed ARSCORER, our final models set the new state-of-the-art results on five widely-used datasets even with fewer decoding steps and lead to completely outperforming their AR counterparts.

In the future, we will extend our explorations to more scenarios since CMLM-based iterative NAR models have been successfully applied in speech and video-related fields (Higuchi et al., 2021). Besides, there is also a need to explore methods for conducting efficient denoising steps for diffusion models (Sohl-Dickstein et al., 2015) since they suffer greatly from low efficiency with numerous denoising steps (Tang et al., 2023; Gong et al., 2023). Lastly, recent advancements in LLMs (Touvron et al., 2023b) hold promise in serving as better scorers for iterative NAR models.

Limitations

Firstly, since CMLM-based iterative NAR models have been applied to various language generation tasks, we only conduct our explorations on machine translation task. Besides, although CMLM-based methods are one of the most widely-used and well-known iterative NAR models, there exist other categories of iterative NAR models, such as editing-based models (Stern et al., 2019; Gu et al., 2019), denoising based models (Lee et al., 2018; Savinov et al., 2021), we only consider CMLM-based methods in this paper. Besides, our proposed efficient strategy, ARSCORER, relies on a pre-trained AR model to serve as a scorer for each token, it brings some extra costs to achieve this AR model and the corresponding prediction confidence.

References

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *ICML*, pages 1403–1413. PMLR.

Xinran Chen, Sufeng Duan, and Gongshen Liu. 2024. Improving non-autoregressive machine translation with error exposure and consistency regularization. *arXiv preprint arXiv:2402.09725*.

Hao Cheng and Zhihua Zhang. 2022. Con-nat: Contrastive non-autoregressive neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6219–6231.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2020. Understanding and improving lexical choice in non-autoregressive translation. In *ICLR*.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and zhaopeng Tu. 2021. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation. In *ACL-IJCNLP*, pages 3431–3441.

Xinwei Geng, Xiaocheng Feng, and Bing Qin. 2021. Learning to rewrite for non-autoregressive neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3297–3308.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6112–6121.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2023. Diffuseq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. *arXiv preprint arXiv:2310.05793*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pages 11181–11191.

Junliang Guo, Linli Xu, and Enhong Chen. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 376–385.

Pei Guo, Yisheng Xiao, Juntao Li, Yixin Ji, and Min Zhang. 2023. Isotropy-enhanced conditional masked language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8278–8289.

Yongchang Hao, Shilin He, Wenxiang Jiao, Zhaopeng Tu, Michael Lyu, and Xing Wang. 2021. Multi-task learning with shared encoder for non-autoregressive machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3989–3996.

Jindřich Helcl, Barry Haddow, and Alexandra Birch. 2022. Non-autoregressive machine translation: It's not as fast as it seems. *arXiv preprint arXiv:2205.01966*.

Yosuke Higuchi, Hirofumi Inaguma, Shinji Watanabe, Tetsuji Ogawa, and Tetsunori Kobayashi. 2021. Improved mask-ctc for non-autoregressive end-to-end asr. In *ICASSP 2021*, pages 8363–8367. IEEE.

Fei Huang, Pei Ke, and Minlie Huang. 2023. [tacl] directed acyclic transformer pre-training for high-quality non-autoregressive text generation. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Fei Huang, Tianhua Tao, Hao Zhou, Lei Li, and Minlie Huang. 2022a. On the learning of non-autoregressive transformers. In *International Conference on Machine Learning*, pages 9356–9376. PMLR.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2022b. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020a. Parallel machine translation with disentangled context transformer. *arXiv preprint arXiv:2001.05136*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2020b. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *ICLR*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*, pages 1317–1327.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Xiaobo Liang, Zecheng Tang, Juntao Li, and Min Zhang. 2023. Open-ended long text generation via masked language modeling. In *ACL*.

Xiaobo Liang, Lijun Wu, Juntao Li, and Min Zhang. 2022. Janus: Joint autoregressive and non-autoregressive training with auxiliary loss for sequence generation. In *EMNLP*, pages 1067–1073.

OpenAI. 2023. Gpt-4 technical report.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 1993–2003.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. 2021. Step-unrolled denoising autoencoders for text generation. *arXiv preprint arXiv:2112.06749*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pages 2256–2265. PMLR.

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

Zecheng Tang, Pinzheng Wang, Keyan Zhou, Juntao Li, Ziqiang Cao, and Min Zhang. 2023. Can diffusion model achieve better performance in text generation? bridging the gap between training and inference! *arXiv preprint arXiv:2305.04465*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. 2022. A survey on non-autoregressive generation for neural machine translation and beyond. *arXiv preprint arXiv:2204.09269*.

Yisheng Xiao, Ruiyang Xu, Lijun Wu, Juntao Li, Tao Qin, Tie-Yan Liu, and Min Zhang. 2023. Amom: Adaptive masking over masking for conditional masked language model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13789–13797.

Pan Xie, Zexian Li, and Xiaohui Hu. 2021. Mvsr-nat: Multi-view subset regularization for non-autoregressive machine translation. *arXiv preprint arXiv:2108.08447*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Details for Follow-up Methods

We supplement the details for follow-up methods of CMLM we adopted for explorations as mentioned in Section 2.

JM-NAT Guo et al. introduce a jointly masked sequence-to-sequence model. Unlike the traditional CMLM which only masks the target sequence during training, JM-NAT also masks the source sequence to help train the encoder more rigorously. Besides, in order to alleviate the problem of translating duplicate words, they propose to train the decoder based on the consecutive masking of the decoder input with an ngram loss function rather than the original uniform masking.

Disco Kasai et al. propose an attention-masking based model, Disentangled Context (DisCo) transformer. During training, Disco is learned to predict each target token given an arbitrary subset of the other reference tokens, which is more efficient than just predicting masked tokens in the original CMLM. During inference, unlike the previous Mask-Predict algorithm which just updates masked tokens in each decoding step (i.e., predicting Y_{mask} based on Y_{obs}), Disco introduces an easy-first policy which where each token will be predicted in each step dependent on relatively easier tokens (i.e., predicting each Y_i based on $Y_{<i}$, where $Y_{<i}$ denotes tokens whose prediction confidence is higher than Y_i in the previous iteration). Disco stops decoding when no new tokens are generated in one specific decoding step. This easy-first policy can largely improving the inference latency.

Multitask-NAT Hao et al. introduces Multitask-NAT which utilizes a shared encoder and separated decoders for both AR and NAR modeling during training. They assume that AR training can bring benefits for NAR training and aim to adopt multi-task learning to transfer the AR knowledge to NAR models through encoder sharing.

RewriteNAT Geng et al. propose RewriteNAT, a new framework that contains a Locator and Revisor module that locate the incorrect words within previously generated translations and then revise them, respectively. Specifically, the Locator module can transform the problem of determining which tokens to be masked in the next decoding step into into a binary classification problem instead of depending on the self-predicted confidence, i.e., the Locator will predict a special symbol (`[MASK]` or `[KEEP]`) for each token. Once the token is predicted as `[MASK]`, it will be masked again, and vice versa. RewriteNAT can finish the generation process once the Locator module predicts all the target tokens as `[KEEP]`.

SMART Ghazvininejad et al. introduce Semi-Autoregressive Training (SMART) to help the training process better match the Mask-Predict algorithm with multiple decoding steps. Specifically, since the model can not see the ground truth tokens during inference, it only takes the model prediction in the previous decoding steps as partially-observed tokens to make predictions. This leads to inconsistency compared with training methods. Thus SMART first constructs a mixed training example and then encourages the model to recover from the model prediction errors during training,

CMLMC Huang et al. propose Conditional Masked Language Model with Correction (CMLMC) which incorporates a self-correction mechanism into traditional CMLM and several modifications on the decoder structure such as exposing the positional encodings and incorporating causal attention layers to differentiate adjacent tokens. CORR is the corresponding variant which only adopts the self-correction mechanism without the structure modifications in CMLMC. Specifically, except for adopting masking methods in target sequence during training, CMLMC aslo replaces the partial unmasked tokens with model predictions based on a fully masked target sequence. Then CMLMC learns to predict the masked tokens and correct the replaced tokens simultaneously during training. During inference, this self-correction mechanism helps the model to correct the unreliable tokens in the unmasked subset.

AMOM Xiao et al. propose an Adaptive Masking Over Masking (AMOM) strategy based on CMLM which contains two different adaptive masking mechanisms which work on the inputs of encoder and decoder respectively. Specifically, based on the ratio of the target sequence, AMOM also masks the specific number of tokens in the source sequence to make the encoder optimization easier. Besides, AMOM conducts an extra masking step where the masking ratio of the target sequence in this step is adaptive to the correction ratio of the model prediction. This two-step masking strategy can help the model capture the masking ratio changes in various decoding steps during inference.

B Training Hyper-parameters

During our experiments, we set training hyper-parameters for CMLM in the same way as CMLM realization in the Fariseq library, and for AMOMC, we follow those adopted in CMLMC (Huang et al., 2022b). Now, we present these training hyper-parameters in Table 6.

Models	Parameters	IWSLT'14 DE→EN	WMT'14 EN↔DE	WMT'16 EN↔RO
CMLM	learning rate	5e-4	7e-4	5e-4
	warmup_step	4k	10k	10k
	dropout	0.3	0.2	0.3
	update_step	300k	300k	300k
AMOMC	learning rate	5e-4	7e-4	5e-4
	warmup_step	30k	40k	15k
	dropout	0.3	0.2	0.3
	update_step	175k	150k	120k

Table 6: Training hyper-parameters for CMLM and AMOMC.