An EM Approach to Non-autoregressive Conditional Sequence Generation

Zhiqing Sun [1] **Yiming Yang** [1]

Abstract

Autoregressive (AR) models have been the dominating approach to conditional sequence generation, but are suffering from the issue of high inference latency. Non-autoregressive (NAR) models have been recently proposed to reduce the latency by generating all output tokens in parallel but could only achieve inferior accuracy compared to their autoregressive counterparts, primarily due to a difficulty in dealing with the multi-modality in sequence generation. This paper proposes a new approach that jointly optimizes both AR and NAR models in a unified Expectation-Maximization (EM) framework. In the E-step, an AR model learns to approximate the regularized posterior of the NAR model. In the M-step, the NAR model is updated on the new posterior and selects the training examples for the next AR model. This iterative process can effectively guide the system to remove the multi-modality in the output sequences. To our knowledge, this is the first EM approach to NAR sequence generation. We evaluate our method on the task of machine translation. Experimental results on benchmark data sets show that the proposed approach achieves competitive, if not better, performance with existing NAR models and significantly reduces the inference latency.

1. Introduction

State-of-the-art conditional sequence generation models (Bahdanau et al., 2014; Gehring et al., 2017; Vaswani et al., 2017) typically rely on an AutoRegressive (AR) factorization scheme to produce the output sequences. Denoting by $x = (x_1, \ldots, x_T)$ an input sequence of length T, and by $y = (y_1, \ldots, y_{T'})$ a target sequence of length T', the conditional probability of y given x is factorized as:

$$p^{AR}(y|x) = \prod_{i=1}^{T'} p(y_i|x, y_1, y_2, \ldots, y_{i-1}). \tag{1}$$

As such a sequential factorization cannot take the full advantage of parallel computing, it yields high *inference latency* as a limitation.

Recently, Non-AutoRegressive (NAR) sequence models (Gu et al., 2017; Lee et al., 2018) are proposed to tackle the problem of inference latency, by removing the sequential dependencies among the output tokens as:

$$p^{NAR}(y|x) = p(T'|x) \prod_{i=1}^{T'} p(y_i|x, T'). \tag{2}$$

This formulation allows each token to be decoded in parallel and hence brings a significant reduction of the inference latency. However, NAR models also suffer from the conditional independence assumption among the output tokens, and usually do not perform as well as their AR counterparts. Such a performance gap is particularly evident when the output distributions exhibit a multi-modality phenomenon (Gu et al., 2017). which means that the input sequence can be mapped to multiple correct output sequences. Such a multi-modal output distribution cannot be represented as the product of conditionally independent distributions for each position in NAR models (See 3.2 for a detailed discussion).

How to overcome the multi-modality issue has been a central focus in recent efforts for improving NAR models. A standard approach is to use sequence-level knowledge distillation (Hinton et al., 2015; Kim & Rush, 2016), which means to replace the target part y of each training instance (x, y) with the system-predicted $\hat{y}$ from a pre-trained AR model (a.k.a. the "teacher model"). Such a replacement strategy removes the one-to-many mappings from the original dataset. The justification for doing so is that in practice we do not really need the sequence generation models to mimic a diverse output distribution for sequence generation tasks such as machine translation[1] and text summarization. Such a knowledge distillation strategy has shown to be effective for improving the performance of NAR models in

[1]Carnegie Mellon University, Pittsburgh, PA 15213 USA. Correspondence to: Zhiqing Sun <zhiqings@cs.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]For example, Google Translate only provide one translation for the input text.

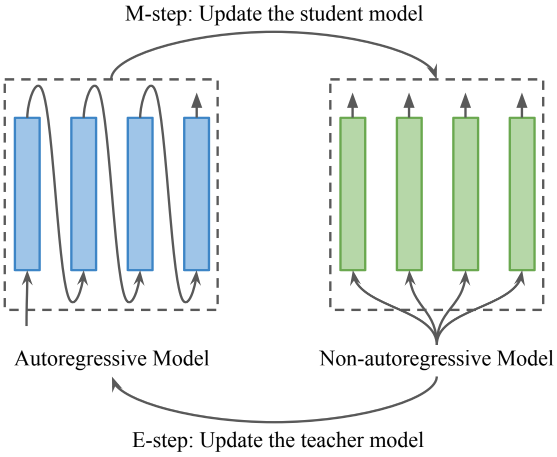
M-step: Update the student model
Autoregressive Model
Non-autoregressive Model
E-step: Update the teacher model

Figure 1. The proposed framework: The AR and NAR models are jointly optimized by alternating between a E-step and a M-step. See Sec. 5.1 for the detailed explanation.

multiple studies (Gu et al., 2017; Kaiser et al., 2018; Li et al., 2019; Ma et al., 2019; Sun et al., 2019; Ghazvininejad et al., 2019; Gu et al., 2019).

We want to point out that in all the NAR methods with knowledge distillation, the teacher AR model is pre-trained only once on ground-truth data and then is used to generate the output training examples for the NAR model. We argue that such a single-pass knowledge distillation process may not be sufficient for optimizing the NAR model as sequence $\hat{y}$ predicted by the AR model cannot be perfect. More importantly, it is not necessarily the best choice for alleviating the multi-modality problem in the NAR model. In other words, without knowing how the choice of $\hat{y}$ by the AR model would effect the training process in the NAR model, the current knowledge distillation approach is unavoidably sub-optimal.

To address this fundamental limitation, we propose a novel Expectation-Maximization (EM) approach to the training of NAR models, where both the teacher (an AR model) and the student (a NAR model) are helping each other in a closed loop, and the iterative updates of the models are guaranteed to converge to a local optimum. This approach gives an extra power to knowledge distillation between AR and NAR models, and is the first EM approach to NAR models, to our knowledge. Fig. 1 illustrates our new framework. In addition, we develop a principled plug-and-play decoding module for effective removal of word duplication in the model's output. Our experiments on three machine translation benchmark datasets show that the proposed approach achieves competitive performance as that of the best NAR models in terms of prediction accuracy, and reduces the inference latency significantly.

2. Related Work

Related work can be divided into two groups, i.e., the non-autoregressive methods for conditional sequence generation, and the various approaches to knowledge distillation in non-autoregressive models.

Recent work on non-autoregressive sequence generation has developed ways to address the multi-modality problem. Several work try to design better training objectives (Shao et al., 2019; Wei et al., 2019) or regularization terms (Li et al., 2019; Wang et al., 2019; Guo et al., 2018). Other methods focus on direct modeling of multi-modal target distributions via hidden variables (Gu et al., 2017; Kaiser et al., 2018; Ran et al., 2019; Ma et al., 2019) or sophisticated output structures (Libovický & Helcl, 2018; Sun et al., 2019). There are also a few recent work (Lee et al., 2018; Stern et al., 2019; Ghazvininejad et al., 2019; Gu et al., 2019) focusing on a multiple-pass iterative-refinement process to generate the final outputs, where the first pass produce an initial output sequence and the following passes refine the sequence iteratively in the inference phase.

As for knowledge distillation in NAR models, Gu et al. (2017) is the first effort to use knowledge distillation (Hinton et al., 2015; Kim & Rush, 2016). Recently, Zhou et al. (2019) analyzed why knowledge distillation would reduce the complexity of datasets and hence be helpful in the training of NAR models. They also used Born-Again networks (BANs) (Furlanello et al., 2018) to produce simplified training data for NAR models.

All the above methods take a pre-trained AR model as the teacher model for knowledge distillation; none of them iteratively updates the teacher model based on the feedback from (or the measured performance of) the NAR model. This is the fundamental difference between existing work and our proposed EM approach in this paper.

3. Problem Definition

3.1. Conditional Sequence Generation

Let us describe the problem of conditional sequence generation in the context of machine translation and use the terms of "sequence" and "sentence", "source" and "input", "target" and "output" interchangeably. We use x and y to denote the source and target sentences, x_i to indicate the i-th token in x, and $\mathcal{X} = \{x^1, x^2, \ldots, x^N\}$ and $\mathcal{Y} = \{y^1, y^2, \ldots, y^N\}$ to be a parallel dataset of N sentence pairs in the source and target languages, respectively.

The training of both AutoRegressive (AR) and Non-AutoRegressive (NAR) sequence generation models is performed via likelihood maximization over parallel data

$(\mathcal{X}, \mathcal{Y})$ as:

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \log p^{AR}(y|x;\phi), \quad (3)$$

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \log p^{NAR}(y|x;\theta), \quad (4)$$

where p^{AR} and p^{NAR} are defined in Eq. 1 and Eq. 2; ϕ and θ are the parameters of the AR and NAR models, respectively.

3.2. Instance-level & Corpus-level Multi-modality

There are two different but not mutually exclusive definitions for the concept of multi-modality (a.k.a., translation uncertainty in machine translation terminology).

Gu et al. (2017) define the multi-modality problem as the existence of one-to-many mappings in the parallel data, which we refer to as the *instance-level* multi-modality. Formally, given the parallel data $(\mathcal{X}, \mathcal{Y})$, if there exist sentences i and j such that $x^i = x^j$ but $y^i \neq y^j$, we say that $(\mathcal{X}, \mathcal{Y})$ contains instance-level multi-modality.

In contrast, Zhou et al. (2019) use the conditional entropy to quantify the translation uncertainty, which we refer as the *corpus-level* multi-modality. It requires the use of an additional alignment model in calculating this quantity. In this paper, we want to avoid the requirement for external alignment tools. Thus, we directly use training-set likelihood of the NAR model as a measure of corpus-level multi-modality. Formally, the Corpus-level Multi-modality (CM) of parallel data $(\mathcal{X}, \mathcal{Y})$ is defined as:

$$\mathrm{CM}_{\mathcal{X}}(\mathcal{Y}) = \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \left[-\log p^{NAR}(y|x;\theta^*) \right], \quad (5)$$

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \log p^{NAR}(y|x;\theta). \quad (6)$$

To make this metric comparable across different data sets, we further define the Normalized Corpus-level Multi-modality (NCM) as:

$$\mathrm{NCM}_{\mathcal{X}}(\mathcal{Y}) = \frac{\mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \left[-\log p^{NAR}_{\mathcal{M}}(y|x;\theta^*) \right]}{\mathbb{E}_{y\sim\mathcal{Y}} \left[|y| \right]} \quad (7)$$

where $|y|$ denotes the length of output sequence y.

4. Properties of NAR Models

How powerful are NAR models? Are they as expressive as AR models? Our answer is both yes and no. On one hand, it is easy to see that NAR models can only capture distributions that can be factorized into conditional independent parts. On the other hand, we will show in the next that if the instance-level multi-modality can be removed from the training data (e.g., via sequence-level knowledge distillation), then NAR models can be as powerful just as AR models.

4.1. Theoretical expressiveness of NAR models

Let us focus on the expressive power of NAR models when the instance-level multi-modality is removed, that is, when there are only one-to-one and many-to-one mappings in training examples. More specifically, we consider the ability of the vanilla Non-Autoregressive Transformer (NAT) models (Gu et al., 2017; Vaswani et al., 2017) in approximating arbitrary continuous $\mathbb{R}^{d\times n} \rightarrow \mathbb{R}^{d\times m}$ sequence-to-sequence single-valued functions, where n and m are the input and output sequence length, and d is the model dimension.

Given the definition of a distance between two functions $f_1, f_2 : \mathbb{R}^{d\times n} \rightarrow \mathbb{R}^{d\times m}$ as:

$$d_p(f_1, f_2) = \left(\int \| f_1(\mathbf{X}) - f_2(\mathbf{X}) \|_p^p d\mathbf{X} \right)^{1/p}. \quad (8)$$

where $p \in [1, \infty)$. We can make the following statement:

Theorem 4.1. *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given continuous sequence-to-sequence function $f : \mathbb{R}^{d\times n} \rightarrow \mathbb{R}^{d\times m}$, there exists a non-autoregressive Transformer network g such that $d_p(f, g) \leq \epsilon$.*

This theorem is a corollary of the Theorem 2 in Yun et al. (2020). For completeness, we provide the formal theorem with proof in the appendix.

4.2. What limits the success of NAR models in practice?

Theorem 4.1 shows that for any sequence-to-sequence dataset containing no instance-level multi-modality, we can always find a good NAT model to fit this dataset. However, in reality, it is still a big challenge for NAT models to fit the distilled deterministic training data very well.

The gap between theory and practice is due to the fact that in theory we may use as many Transformer layers as needed, but in reality, there are only a few layers (e.g., 6 layers) in the Transformer model. This would greatly restrict the model capacity of real NAR models.

To further understand the limitation let us examine the following two hypotheses:

- The NAT model intrinsically cannot accurately produce very long output sequences when it has only a few Transformer layers.
- The corpus-level multi-modality in data makes it hard for NAT models to deal with (i.e., to memorize the "mode" in the output for each input).

These hypotheses focus on two different reasons that might cause the poor performance of NAR models. In order to

Table 1. Toy examples illustrating the two types of synthetic experiments.

	source		target
Experiment I	2 1 4 3	→	2 2 1 4 4 4 4 3 3 3
	2 2 3	→	2 2 2 2 3 3 3
	2 1 5	→	2 2 1 5 5 5 5
Experiment II	2 1 4 3	→	0 2 2 1 4 4 4 4 3 3 3 0 0 0
	2 2 3	→	0 0 0 2 2 2 2 3 3 3 0
	2 1 5	→	2 2 1 5 5 5 5 5 0 0 0 0

Table 2. The accuracy in whole-sentence matching of the AR and NAR models over 1000 synthetic examples.

	Experiment I		Experiment II	
Models	AR	NAR	AR	NAR
Accuracy(%)	99.9	95.7	99.8	00.0

verifying which one is true, we design two types of synthetic data and experiments.

In Experiment I, a synthetic translation dataset is constructed as follows: The source and target sides share the same vocabulary of $\{1, 2, 3, 4, 5\}$. 1, 2, 3, and 4, and 5 are translated into 1, 2 2, 3 3 3, 4 4 4 4, and 5 5 5 5 5, respectively. The translation is deterministic, i.e., no multi-modality in the resulted parallel dataset. In Experiment II, we randomly insert four 0 in the front or the back of each target sentence in the 1st dataset. In other words, the source-to-target translation in the 2nd dataset is non-deterministic and hence is with corpus-level multi-modality. In addition, we filter the source data in Experiment II to make sure that there is no instance-level multi-modality in this dataset. The toy examples illustrating the two types of datasets can be found in Tab. 1. Following such rules we randomly generated 2,000,000 sentences for training and 1000 sentences for testing; both the training and testing *source* sentences have the length of 30.

We trained both the AR transformer and the NAR Transformer on these synthetic datasets, each of which consists of a 3-layer encoder and a 3-layer decoder. The detailed model settings can be found in the appendix. In our evaluation we used the ground-truth lengths for the decoding of both the AR and NAR Transformers; the whole-sentence matching accuracy of those models are listed in Tab. 2.

The results in Experiment I show that both the autoregressive Transformer and the non-autoregressive Transformer can achieve an high accuracy of 99.9% and 95.7%, respective, when the training data do not have the multi-modality. In contrast, the results of Experiment II show that the non-autoregressive Transformer model failed completely on the synthetic dataset with corpus-level multi-modality. The sharp contrast in these synthetic experiments indicates that the real problem with NAR models is indeed due to the corpus-level multi-modality issue.

5. Proposed Method

Let us formally introduce our EM approach to addressing the multi-modality issue in NAR models, followed by a principled decoding module for effective removal of word duplication in the predicted output.

5.1. The EM Framework

With the definition of the corpus-level multi-modality (i.e., CM in Eq. 5), we consider how to reduce this quantity for the better training of NAR models. Formally, given source data $\mathcal{X}$, we want to find target data $\mathcal{Y}^*$ that satisfies the following property:

$$\begin{aligned}\mathcal{Y}^* &= \arg\min_{\mathcal{Y}} \mathrm{CM}_{\mathcal{X}}(\mathcal{Y}) \\ &= \arg\min_{\mathcal{Y}} \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \left[-\log p^{NAR}(y|x;\theta^*)\right].\end{aligned}$$

However, there can be many trivial solutions for $\mathcal{Y}^*$. For example, we can simply construct a dataset with no variation in the output to achieve zero corpus-level multi-modality. To avoid triviality, we may further apply a constraint to $\mathcal{Y}^*$. This leads us to the posterior regularization framework.

Posterior Regularization (Ganchev et al., 2010) is a probabilistic framework for structured, weakly supervised learning. In this framework, we can re-write our objective as following:

$$\mathcal{L}_{\mathcal{Q}}(\theta) = \min_{q\in\mathcal{Q}} \mathbb{KL}(q(\mathcal{Y})\|p^{NAR}(\mathcal{Y}|\mathcal{X};\theta)),$$

where q is the posterior distribution of $\mathcal{Y}$ and $\mathcal{Q}$ is a constraint posterior set that controls the quality of the parallel data given by:

$$\mathcal{Q} = \{q(\mathcal{Y}) : \mathbb{E}_{\mathcal{Y}\sim q}\left[\mathrm{Q}_{\mathcal{X}}(\mathcal{Y})\right] \geq \mathbf{b}\},$$

where Q is a metric for quality mapping from $(\mathcal{X}, \mathcal{Y})$ to $\mathbb{R}^N$ in the training set and $\mathbf{b}$ is a bound vector.

For sequence generation tasks, there are many corpus-level quality metrics, such as BLEU (Papineni et al., 2002) and ROUGE (Lin & Hovy, 2003). However, they are known to be inaccurate for measuring the quality of single sentence pairs. Thus, we use the likelihood score of a pre-trained AR model as a more reliable quality metric:

$$[\mathrm{Q}_{\mathcal{X}}(\mathcal{Y})]_i = \mathrm{Q}_{x^i}(y^i) = \log p^{AR}(y^i|x^i;\phi^1),$$

where ϕ^1 denotes that the AR model trained on the original ground-truth dataset.

Given the posterior regularization likelihood $\mathcal{L}_Q(\theta)$, we use the EM algorithm (McLachlan & Krishnan, 2007; Ganchev et al., 2010) to optimize it. In the E-step (a.k.a,the inference procedure), the goal is to fix p^{NAR} and update the posterior distribution:

$$q^{t+1} = \arg\min_{q \in Q} \mathbb{KL}(q(\mathcal{Y}) \| p^{NAR}(\mathcal{Y}|\mathcal{X};\theta^t)),$$

In the M-step (a.k.a., learning procedure), we will fix $q(\mathcal{Y})$ and update θ to maximize the expected log-likelihood:

$$\theta^{t+1} = \arg\max_{\theta} \mathbb{E}_{q^{t+1}}[\log p^{NAR}(\mathcal{Y}|\mathcal{X};\theta)],$$

Next, we introduce the details of the E-step and the M-step in our framework.

Inference Procedure The E-step aims to compute the posterior distribution $q(\mathcal{Y})$ that minimizes the KL divergence between $q(\mathcal{Y})$ and $p^{NAR}(\mathcal{Y}|\mathcal{X})$. Ganchev et al. (2010) show that for graphical models, $q(\mathcal{Y})$ can be efficiently solved in its dual form. Specifically, the primal solution q^* is given in terms of the dual solution $\boldsymbol{\lambda}^*$ by:

$$\begin{aligned} q^*(\mathcal{Y}) &\propto p^{NAR}(\mathcal{Y}|\mathcal{X};\theta^t)\exp\{\boldsymbol{\lambda}^* \cdot \mathrm{Q}_{\mathcal{X}}(\mathcal{Y})\} \\ &\propto \prod_{i=1}^{N} \left[p^{NAR}(y^i|x^i;\theta^t) \left(p^{AR}(y^i|x^i;\phi^0)\right)^{\lambda_i^*} \right], \end{aligned}$$

However, a problem here, as pointed out by Zhang et al. (2018), is that it is hard to specify the hyper-parameter $\mathbf{b}$ to effectively bound the expectation of the features for neural models. Besides, even when $\mathbf{b}$ is given, calculating $\boldsymbol{\lambda}^*$ is still intractable for neural models. Therefore, in this paper, we introduce another way to compute $q(\mathcal{Y})$.

We first factorize $q(\mathcal{Y})$ as the product of $\{q(y^i)\}$, and then follow the idea of amortized inference (Gershman & Goodman, 2014) to parameterize $q(y^i)$ with an AR sequence generation model:

$$q(\mathcal{Y}) = \prod_{i=1}^{N} p^{AR}(y^i|x^i;\phi).$$

The E-step can thus be re-written as follows:

$$\phi^{t+1} = \arg\min_{\phi \in \mathcal{Q}'} \mathbb{E}_{x\sim\mathcal{X}} \mathbb{KL}(p^{AR}(y|x;\phi) \| p^{NAR}(y|x;\theta^t)).$$

where the new constraint posterior set $\mathcal{Q}'$ is defined as

$$\left\{\phi : \mathbb{E}_{p^{AR}(\mathcal{Y}|\mathcal{X};\phi)}\left[\mathrm{CQ}_{\mathcal{X}}(\mathcal{Y})\right] \geq b\right\}.$$

We further apply the REINFORCE algorithm (Williams, 1992) to estimate the gradient of $\mathcal{L}_{\mathcal{Q}}$ w.r.t. $\phi \in \mathcal{Q}'$:

$$\begin{aligned} &\nabla_\phi \mathcal{L}_{\mathcal{Q}} = \mathbb{E}_{x\sim\mathcal{X}}\mathbb{E}_{y\sim p^{AR}(y|x;\phi)} \\ &\left(-\log \frac{p^{NAR}(y|x;\theta^t)}{p^{AR}(y|x;\phi)} \nabla_\phi \log p^{AR}(y|x;\phi) \right). \end{aligned}$$

This can be intuitively viewed as to construct a weighted pseudo training dataset $(\mathcal{X}^{t+1}, \mathcal{Y}^{t+1})$, where the training examples are sampled at random from $p^{AR}(y|x;\phi)$ and weighted by $\log \frac{p^{NAR}(y|x;\theta^t)}{p^{AR}(y|x;\phi)}$.

In practice, we find that there are two problems in implementing this algorithm: One is that sampling from $p^{AR}(y|x;\phi)$ is very inefficient; the other is that the constraint $\phi \in \mathcal{Q}'$ cannot be guaranteed. Therefore, we instead use a heuristic way when constructing the pseudo training dataset $(\mathcal{X}^{t+1}, \mathcal{Y}^{t+1})$: We first follow Wu et al. (2018) to replace the inefficient sampling process with beam search (Sutskever et al., 2014) on $p^{AR}(y|x;\phi^t)$, and then filter out the candidates that doesn't satisfy the following condition:

$$\mathrm{Q}_x(y) \geq \hat{b}_i,$$

where $\hat{b}_i$ is a newly introduced pseudo bound that can be empirically set by early stopping. In this way, we control the quality of $p^{AR}(y|x;\phi^{t+1})$ by manipulating the quality of its training data. Finally, we choose the ones with the highest $p^{AR}(y|x;\phi^t)\log\frac{p^{NAR}(y|x;\theta^t)}{p^{AR}(y|x;\phi^t)}$ score as the training examples in $\mathcal{Y}^{t+1}$, and $\mathcal{X}^{t+1}$ is merely a copy of $\mathcal{X}$.

In each E-step, in principle, we should let ϕ^{t+1} converge under the current NAR model. Although this can be achieved by constructing the pseudo datasets and train the AR model for multiple times, it is practically prohibitive due to the expensive training cost. We, therefore, use only a single update iteration of the AR model in the inner loop of each E-step.

Learning Procedure In the M-step, we seek to learn the parameters θ^{t+1} with the parameterized posterior distribution $p^{AR}(\mathcal{Y}|\mathcal{X};\phi^{t+1})$. However, directly sampling training examples from the AR model will cause the instance-level multi-modality problem. Therefore, we apply sequence-level knowledge distillation (Kim & Rush, 2016) to solve this problem, that is, we only use the targets with maximum likelihood in the AR model to train the NAR model:

$$\theta^{t+1} = \arg\max_{\theta} \mathbb{E}_{(x,y)\sim(\mathcal{X},\hat{\mathcal{Y}}^{t+1})} \log p^{NAR}(y|x;\theta).$$

where $\hat{\mathcal{Y}}^{t+1}$ denotes the training examples produced by the AR model $p^{AR}(\mathcal{Y}|\mathcal{X};\phi^{t+1})$.

Joint Optimization We first pre-train an AR teacher model on the ground-truth parallel data as $p^{AR}(y|x;\phi^1)$. Then we alternatively optimize p^{NAR} and p^{AR} until convergence. We summarize the optimization algorithm in Alg. 1. In our EM method, the AR and NAR models are jointly optimized to reduce the corpus-level multi-modality.

Algorithm 1 An EM approach to NAR models

```
Input: Parallel training dataset $(\mathcal{X}, \mathcal{Y})$
$t = 0$
Pre-train $p^{AR}(y|x; \phi^1)$ on $(\mathcal{X}, \mathcal{Y})$.
while not converged do
  $t = t + 1$
  ⊡ M-Step: Learning Procedure
  Construct the distillation dataset $\hat{\mathcal{Y}}^t$ with $p^{AR}(y|x; \phi^t)$.
  Train $p^{NAR}(y|x; \theta^t)$ on $(\mathcal{X}, \hat{\mathcal{Y}}^t)$.
  ⊡ E-Step: Inference Procedure
  Construct the pseudo dataset $(\mathcal{X}^{t+1}, \mathcal{Y}^{t+1})$.
  Train $p^{AR}(y|x; \phi^{t+1})$ on $(\mathcal{X}^{t+1}, \mathcal{Y}^{t+1})$.
end while
Output: A NAR model $p^{NAR}(y|x; \theta^t)$.
```

5.2. The Optimal De-duplicated Decoding Module

Word duplication is a well-known problem in NAR models caused by the multi-modality issue. To improve the performance of NAR models, some previous work (Lee et al., 2018; Li et al., 2019) remove any duplication in the model prediction by collapsing multiple consecutive occurrences of a token. Such an empirical approach is not technically sound. After collapsing, the length of the target sequence is changed. This will cause a discrepancy between the predicted target length and the actual sequence length and thus make the final output sub-optimal.

We aim to solve the word duplication problem in NAR models, while preserving the original sequence length. Similar to Sun et al. (2019), we use the Conditional Random Fields (CRF) model (Lafferty et al., 2001) for the decoding of NAR models. The CRF model is manually constructed as follows. It treats the tokens to be decoded as the predicted labels. The unitary scores of the labels in each position are set to be NAR models' output distribution and the transition matrix is set to $-\infty \cdot \mathbf{I}$, where $\mathbf{I}$ is an identity matrix.

Our model is able to find the optimal decoding when considering only the top-3 candidates w.r.t. the unitary scores in each position:

Proposition 5.1. *In a CRF with a transition matrix of* $-\infty \cdot \mathbf{I}$*, only top-3 likely labels for each position are possible to appear in the optimal (most likely) label sequence.*

We can thus crop the transition matrix accordingly by only keeping a 3×3 transition sub-matrix between each pair of adjacent positions. The forward-backward algorithm (Lafferty et al., 2001) is then applied on the top-3 likely labels and the 3×3 transition sub-matrices to find the optimal decoding with the linear time complexity of $O(|y|)$.

The proposed decoding module is a lightweight plug-and-play module that can be used for any NAR models. Since this principled decoding method is guaranteed to find the optimal prediction that has no word duplication, we refer it as optimal de-duplicated (ODD) decoding method[2].

6. Experiments

6.1. Experimental Settings

We use several benchmark tasks to evaluate the effectiveness of the proposed method, including IWSLT14[3] German-to-English translation (IWSLT14 De-En) and WMT14[4] English-to-German/German-to-English translation (WMT14 En-De/De-En). For the WMT14 dataset, we use Newstest2014 as test data and Newstest2013 as validation data. For IWSLT14/WMT14 datasets, we split words into BPE tokens (Sennrich et al., 2015), forming a 10k/32k vocabulary shared by source and target languages.

We use the Transformer (Vaswani et al., 2017) model as the AR teacher, and the vanilla Non-Autoregressive Transformer (NAT) (Gu et al., 2017) model with sequence-level knowledge distillation (Kim & Rush, 2016) as the NAR baseline. For both AR and NAR models, we use the original `base` setting for the WMT14 dataset, and a `small` setting for the IWSLT14 dataset. To investigate the influence of the model size on our method, we also evaluate `large`/`base` NAT models on WMT14/IWSLT14 datasets as a larger model setting. These larger NAT models are not used in the EM iterations. They are merely trained with the final AR teacher from the EM iterations of the original model (`base`/`small` for WMT14/IWSLT14). The detailed settings of the model architectures can be founded in the appendix.

We use Adam optimizer (Kingma & Ba, 2014) and employ a label smoothing (Szegedy et al., 2016) of 0.1 in all experiments. The `base` and `large` models are trained for 125k steps on 8 TPUv3 nodes in each iteration, while the `small` models are trained for 20k steps. We use a beam size of 20/5 for the AR model in the M/E-step of our EM training algorithm. The pseudo bounds $\{\hat{b}_i\}$ is set by early stopping with the accuracy on the validation set.

6.2. Inference

During decoding, the target length $l = |y|$ is predicted by an additional classifier conditional on the source sentence: $l = \arg\max_{T'} p(T'|x)$. We can also try different target

[2] Although this method does not allow any word duplication in the output sequence, it is still able to produce any sequence. To solve the problem that multiple consecutive occurrences of a token cannot be captured by our decoding method, we can introduce a special "⟨concat⟩" symbol. For example, "very very good" can be represented by "very ⟨concat⟩ very good".

[3] https://wit3.fbk.eu/

[4] http://statmt.org/wmt14/translation-task.html

Table 3. Performance of BLEU score on WMT14 En-De/De-En and IWSLT14 De-En tasks for single-pass NAR models. "/" denotes that the results are not reported in the original paper. Transformer (Vaswani et al., 2017) results are based on our own reproduction.

Models	WMT14		IWSLT14	Latency	Speedup
	En-De	De-En	De-En		
Autoregressive teacher model					
Transformer w/ beam size 5	27.84	32.14	34.69	393ms	1.00×
Non-autoregressive models					
NAT-FT	17.69	21.47	/	39$ms^\dagger$	15.6×†
LT	19.80	/	/	105$ms^\dagger$	/
ENAT	20.65	23.02	24.13	24$ms^\dagger$	25.3×†
NAT-BAN	21.47	/	/	/	/
NAT-REG	20.65	24.77	23.89	22$ms^\dagger$	27.6×†
NAT-HINT	21.11	25.24	25.55	26$ms^\dagger$	30.2×†
NAT-CTC	17.68	19.80	/	350$ms^\dagger$	3.42×†
FlowSeq-base	21.45	26.16	27.55	/	/
ReorderNAT	22.79	27.28	/	/	16.1×†
NAT-CRF	23.44	27.22	27.44	37$ms^\dagger$	10.4×†
Ours					
NAT baseline	19.55	23.44	22.35	22ms	17.9×
+ EM training	23.27	26.73	29.38	22ms	17.9×
+ ODD decoding	**24.54**	**27.93**	**30.69**	24ms	16.4×

Table 4. Performance of BLEU score on WMT14 En-De/De-En and IWSLT14 De-En tasks for NAR models with rescoring or iterative refinement. "/" denotes that the results are not reported in the original paper. Transformer (Vaswani et al., 2017) results are based on our own reproduction.

Models	WMT14		IWSLT14	Latency	Speedup
	En-De	De-En	De-En		
Autoregressive teacher model					
Transformer w/ beam size 5	27.84	32.14	34.69	393ms	1.00×
Non-autoregressive models					
NAT-FT (rescore 10)	18.66	22.41	/	79$ms^\dagger$	7.68×†
LT (rescore 10)	21.00	/	/	/	/
ENAT (rescore 9)	24.28	26.10	27.30	49$ms^\dagger$	12.4×†
NAT-REG (rescore 9)	24.61	28.90	28.04	40$ms^\dagger$	15.1×†
NAT-HINT (rescore 9)	25.20	29.52	28.80	44$ms^\dagger$	17.8×†
FlowSeq base (rescore 15)	23.08	28.07	/	/	1.04×†
FlowSeq large (rescore 15)	25.03	30.48	/	/	0.96×†
ReorderNAT (rescore 7)	24.74	29.11	/	/	7.40×†
NAT-CRF (rescore 9)	26.07	29.68	29.99	63$ms^\dagger$	6.14×†
NAT-IR (10 iterations)	21.61	25.48	/	222$ms^\dagger$	1.88×†
CMLM (4 iterations)	25.94	29.90	/	/	3.05×†
CMLM (10 iterations)	27.03	**30.53**	/	/	1.30×†
LevT (7+ iterations)	**27.27**	/	/	/	3.55×†
Ours					
NAT baseline (rescore 9)	22.24	26.21	29.64	41ms	9.59×
+ EM training	25.03	28.78	31.74	41ms	9.59×
+ ODD decoding	25.75	29.29	32.66	43ms	9.14×
+ larger model	26.21	29.81	**33.25**	65ms	6.05×

lengths ranging from $(l-b)$ to $(l+b)$ and obtain multiple translations with different lengths, where b is the half-width, and then use the AR model $p^{AR}(y|x;\phi^1)$ as the rescorer to select the best translation. Such a decoding and rescoring process can be conducted in parallel and is referred as parallel length decoding. To make a fair comparison with previous work, we set b to 4 and use 9 candidate translations for each sentence. For each dataset, we evaluate the model performance with the BLEU (Papineni et al., 2002) score. [5] We evaluate the average per-sentence decoding latency[6] on WMT14 En-De test sets with batch size 1 on a single NVIDIA GeForce RTX 2080 Ti GPU by averaging 5 runs.

6.3. Main Results

We compare our model with the Transformer (Vaswani et al., 2017) teacher model and several NAR baselines, including NAT-FT (Gu et al., 2017), LT (Kaiser et al., 2018), ENAT (Guo et al., 2018), NAT-BAN (Zhou et al., 2019), NAT-REG (Wang et al., 2019), NAT-HINT (Li et al., 2018), NAT-CTC (Libovickỳ & Helcl, 2018), Flowseq (Ma et al., 2019), ReorderNAT (Ran et al., 2019), NAT-CRF (Sun et al., 2019), NAT-IR (Lee et al., 2018), CMLM (Ghazvininejad et al., 2019), and LevT (Gu et al., 2019).

Tab. 3 provides the performance of our method with maximum likely target length l, together with other NAR baselines that generate output sequences in a single pass. From the table, we can see that the EM training contributes most to the improvement of the performance. The optimal deduplicated (ODD) decoding also significantly improves the model performance. Compared with other models, our method significantly outperforms all of them, with nearly no additional overhead compared with the vanilla NAT.

Tab. 4 illustrates the performance of our method equipped with rescoring and other baselines equipped with rescoring or iterative refinement. Since our method has nearly no additional overhead compared with the vanilla NAT, to make a fair comparison with previous work (Kaiser et al., 2018; Lee et al., 2018; Ma et al., 2019; Sun et al., 2019; Gu et al., 2019), we also show the results of our method with a larger model setting. From the table, we can still find that the EM training significantly improves the performance of the vanilla NAT model, but the effect of the OOD decoding is not as significant as in the single-pass setting. This shows that the rescoring process can mitigate the word duplication problem to some extent. Surprisingly, we also find that using the larger model also does not bring much gain. A potential explanation for this is that since our EM algorithm significantly simplifies the training dataset and the NAT model can be over-parameterized, there is no much gain in further increasing the model size. Compared with

[5] We follow common practice in previous works to make a fair comparison. Specifically, we use tokenized case-sensitive BLEU for WMT datasets and case-insensitive BLEU for IWSLT datasets.

[6] † in Tab. 3 and Tab. 4 indicates that the latency and the speedup may be affected by hardware settings and are thus not fair for direct comparison.

other baselines, our method significantly outperforms these rescored single-pass NAR methods and achieves competitive performance to iterative-refinement models with a much better speedup. Note that these iterative-refinement models (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019) still rely on sequence-level knowledge distillation in training, which indicates that it is still a hard problem for these approaches to capture multi-modality in the real data. Our EM algorithm may further improve their performance. We leave combining the two techniques for future work.

6.4. Analysis of the Convergence

We analyze the convergence of the EM algorithm. We show the dynamics of the performance of the NAR model (test BLEU), the performance of the AR model (test BLEU), and the Normalized Corpus-level Multi-modality (NCM, defined by Eq. 7) on the WMT14 En-De dataset. The results are shown in Tab. 5.

We describe the detailed optimization process here to clarify how our EM algorithm works with early stopping in this example. In the first 5 iterations, as we have no idea how to set $\{\hat{b}^i\}$ precisely, we simply set them to zeros. But after the 5^{th} iteration, we find an accuracy drop in the validation set. So we will re-use the quality metrics at the 4^{th} iteration to set $\{\hat{b}^i\}$ and continue the EM algorithm until convergence.

We can see that our EM algorithm takes only a few iterations to convergence, which is very efficient. With the EM algorithm continues, the NCM metric, which can be regarded as the optimization objective, decreases monotonically. The performance of the NAR model and the performance of the AR model also converge after 5 iterations.

Table 5. Analysis of the convergence for the EM algorithm on WMT14 En-De test BLEU. NCM represents the Normalized Corpus-level Multi-modality. All the models are evaluated without ODD decoding and rescoring. Iteration $t = 1, 2, 3, 4, 5^*$ are performed without quality constraints. Iteration $t = 5, 6$ are re-started from $t = 4$ with quality constraints.

	NAR Model	AR Model	NCM
$t = 1$	19.55	**27.84**	2.88
$t = 2$	22.27	27.50	2.33
$t = 3$	22.85	27.13	2.24
$t = 4$	**23.27**	26.78	2.16
$t = 5^*$	22.86	26.11	**2.04**
$t = 5$	23.18	26.72	2.12
$t = 6$	23.16	26.75	2.11

6.5. Analysis of the Amortized Inference

In our EM method, we employ amortized inference and parametrize the posterior distribution of the target sequences by an AR model. In this section, we investigate the importance of amortized inference. Specifically, we try to directly train the NAR model on $(\mathcal{X}^{t+1}, \mathcal{Y}^{t+1})$ in the M-step. The results are shown in Tab. 6. We can see that parameterizing the posterior distribution by a unified AR model always improves the performance of the NAR model.

Table 6. Analysis of the amortized inference for iteration $t = 2, 3, 4$ on WMT En-De test BLEU. All the models are evaluated without ODD decoding and rescoring. We show the results of the NAR models using different training data in the M-step.

	amortized	non-amortized
$t = 2$	**22.27**	21.78
$t = 3$	**22.85**	22.44
$t = 4$	**23.27**	22.98

6.6. Analysis of the Optimal De-duplicated Decoding

Finally, we analyze the effect of the proposed optimal de-duplicated (ODD) decoding approach. We compare it with another plug-and-play de-duplicated decoding approach, that is, “removing any repetition by collapsing multiple consecutive occurrences of a token” (Lee et al., 2018), which we refer to as post-de-duplication. The results are shown in Tab. 7. We can see that the proposed ODD decoding method consistently outperforms this empirical method. This shows that our proposed method can overcome the sub-optimal problem of the post-de-duplication method.

Table 7. Analysis of the ODD decoding on WMT En-De test BLEU. All the models are trained with our EM algorithm.

	WMT En-De	WMT De-En	IWSLT
post-de-duplication	23.67	26.93	24.96
+ rescoring 9	25.56	28.92	32.03
ODD decoding	**24.54**	**27.93**	**30.69**
+ rescoring 9	**25.75**	**29.29**	**32.66**

7. Conclusion

This paper proposes a novel EM approach to non-autoregressive conditional sequence generation, which effectively addresses the multi-modality issue in NAR training by iterative optimizing both the teacher AR model and the student NAR model. We also developed a principled plug-and-play decoding method for efficiently removing word duplication in the model's output. Experimental results on three tasks prove the effectiveness of our approach. For future work, we plan to examine the effectiveness of our method in a broader range of applications, such as text summarization.

Acknowledgements

We would like to thank Meng Qu and anonymous reviewers for valuable comments and suggestions. This work is supported in part by the National Science Foundation (NSF) under grant IIS-1546329.

References

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

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

Ganchev, K., Gillenwater, J., Taskar, B., et al. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

Gershman, S. and Goodman, N. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6114–6123, 2019.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Gu, J., Wang, C., and Zhao, J. Levenshtein transformer. *arXiv preprint arXiv:1905.11006*, 2019.

Guo, J., Tan, X., He, D., Qin, T., Xu, L., and Liu, T.-Y. Non-autoregressive neural machine translation with enhanced decoder input. *arXiv preprint arXiv:1812.09664*, 2018.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kaiser, Ł., Roy, A., Vaswani, A., Parmar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*, 2018.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

Lee, J., Mansimov, E., and Cho, K. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*, 2018.

Li, Z., He, D., Tian, F., Qin, T., Wang, L., and Liu, T.-Y. Hint-based training for non-autoregressive translation. 2018.

Li, Z., Lin, Z., He, D., Tian, F., Qin, T., Wang, L., and Liu, T.-Y. Hint-based training for non-autoregressive machine translation. *arXiv preprint arXiv:1909.06708*, 2019.

Libovickỳ, J. and Helcl, J. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. *arXiv preprint arXiv:1811.04719*, 2018.

Lin, C.-Y. and Hovy, E. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 150–157, 2003.

Ma, X., Zhou, C., Li, X., Neubig, G., and Hovy, E. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*, 2019.

McLachlan, G. J. and Krishnan, T. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Ran, Q., Lin, Y., Li, P., and Zhou, J. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*, 2019.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Shao, C., Zhang, J., Feng, Y., Meng, F., and Zhou, J. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. *arXiv preprint arXiv:1911.09320*, 2019.

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*, 2019.

Sun, Z., Li, Z., Wang, H., He, D., Lin, Z., and Deng, Z. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, pp. 3011–3020, 2019.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, Y., Tian, F., He, D., Qin, T., Zhai, C., and Liu, T.-Y. Non-autoregressive machine translation with auxiliary regularization. *arXiv preprint arXiv:1902.10245*, 2019.

Wei, B., Wang, M., Zhou, H., Lin, J., and Sun, X. Imitation learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1906.02041*, 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wu, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866*, 2018.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.

Zhang, J., Liu, Y., Luan, H., Xu, J., and Sun, M. Prior knowledge integration for neural machine translation using posterior regularization. *arXiv preprint arXiv:1811.01100*, 2018.

Zhou, C., Neubig, G., and Gu, J. Understanding knowledge distillation in non-autoregressive machine translation. *arXiv preprint arXiv:1911.02727*, 2019.